CCE60220

Perangkat Bergerak (TKOM)

Fakultas Ilmu Komputer Universitas Brawijaya

MATAKULIAH	: **Perangkat Bergerak (TKOM)**
KODE/ STATUS	: CCE60220
SKS	: 2
Dosen	: Dahnial Syauqy, S.T, M.T
Email	: dahnial87@ub.ac.id
Ruang	:

Agenda Perkuliahan

1. Intro dan overview perkuliahan
2. Sejarah dan perkembangan teknologi perangkat bergerak
3. Komponen perangkat keras dan perangkat lunak
4. Pengenalan dan instalasi android studio serta aplikasi sederhana
5. Intent dan passing data pada Android Studio
6. Android Studio: Sensor reading
7. Android Studio: Storage & shared preference
8. **========================UTS**
9. Pengenalan dan aplikasi sederhana dengan MIT AppInventor
10. Appinventor: variable, looping, conditional, tinyDB, file
11. appInventor: sensor reading & persiapan project
12. Appinventor: Akuisisi gambar dan suara
13. Appinventor: komunikasi bluetooth
14. Appinventor: basic animation
15. Presentasi kelompok
16. **========================UAS**

Preparation

What you need

- A Computer

What you should already know

- JAVA Programming Language

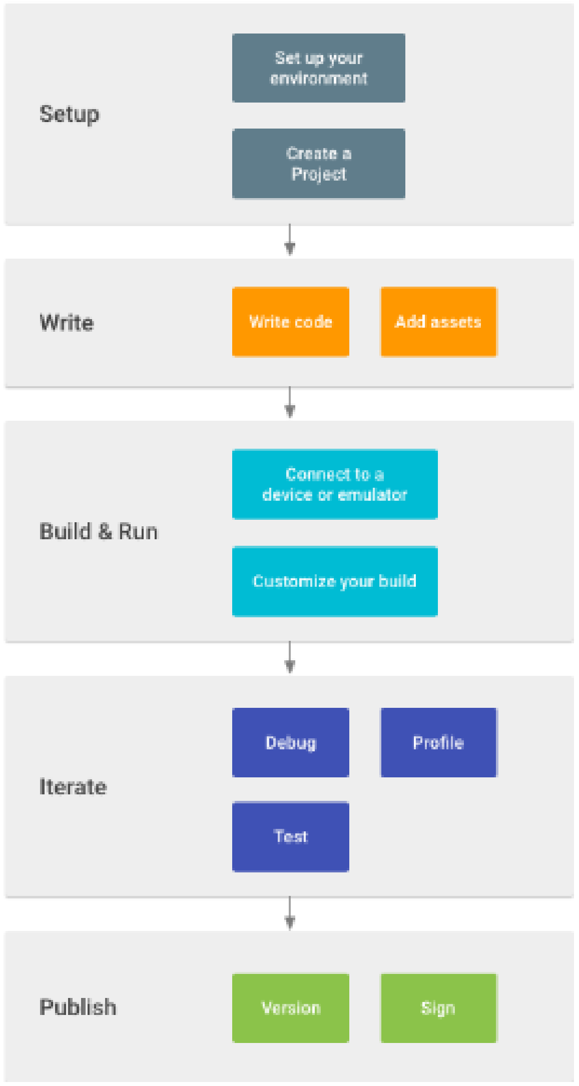
Software Requirements

- Java Development Kit (JDK)
- Android Studio with SDK

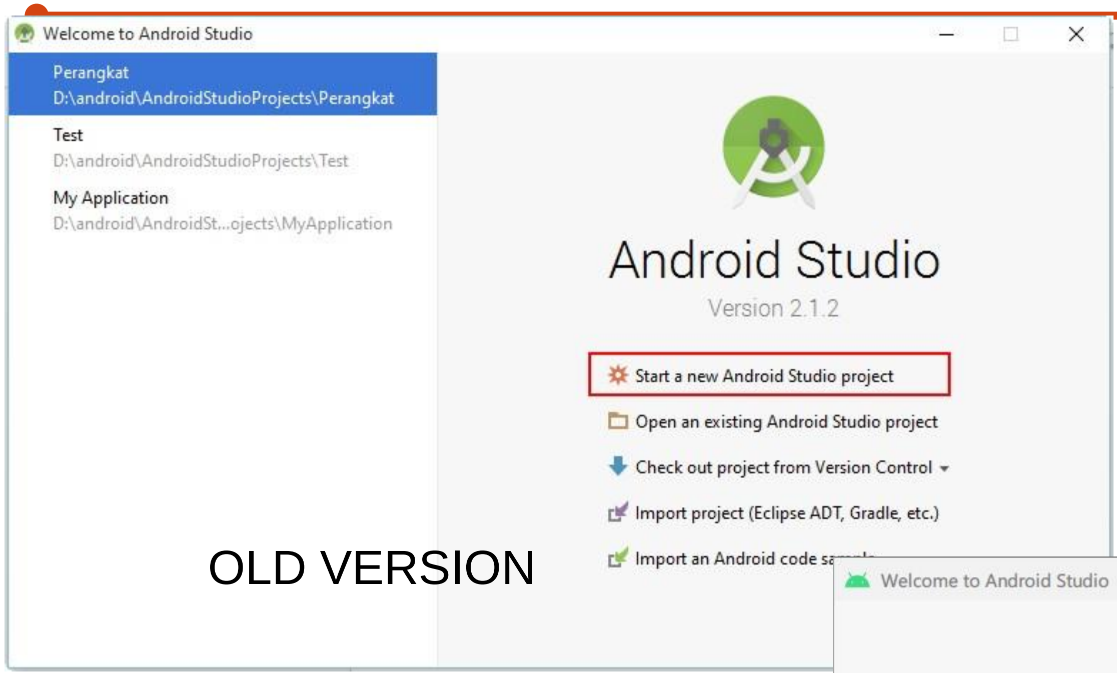
Have you prepared all of them?

Workflow Basics

1. Set up your workspace
2. Write your app
3. Build and run
4. Debug, profile, and test
5. Publish



Create your first app


Welcome to Android Studio

Perangkat
D:\android\AndroidStudioProjects\Perangkat

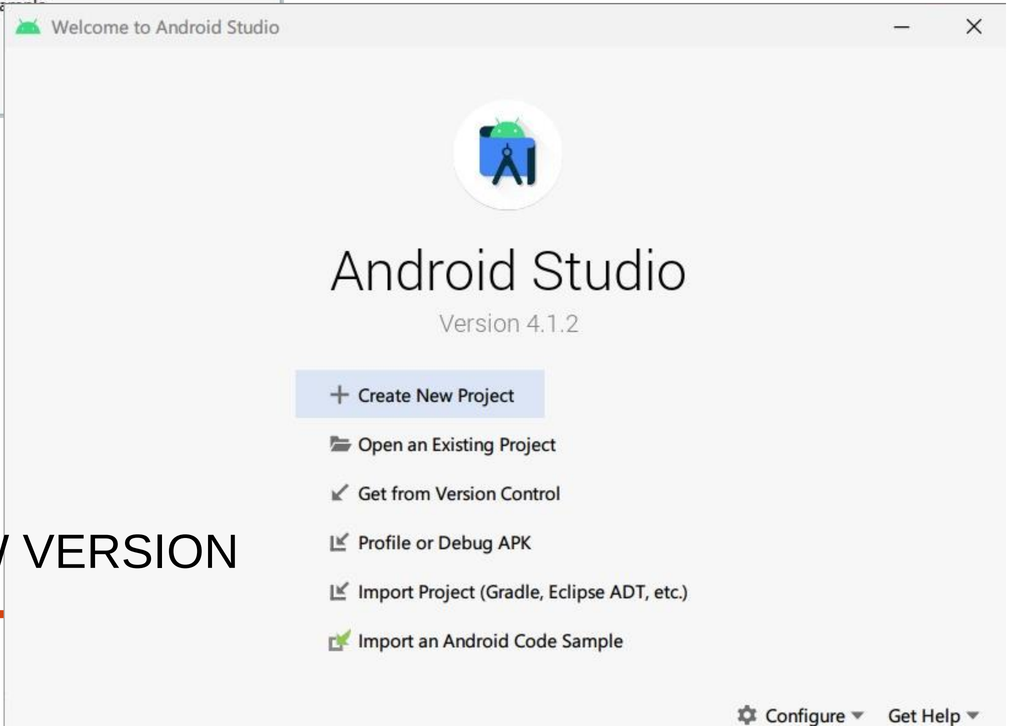
Test
D:\android\AndroidStudioProjects\Test

My Application
D:\android\AndroidSt...ojects\MyApplication

Android Studio
Version 2.1.2

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 sa

OLD VERSION


Welcome to Android Studio

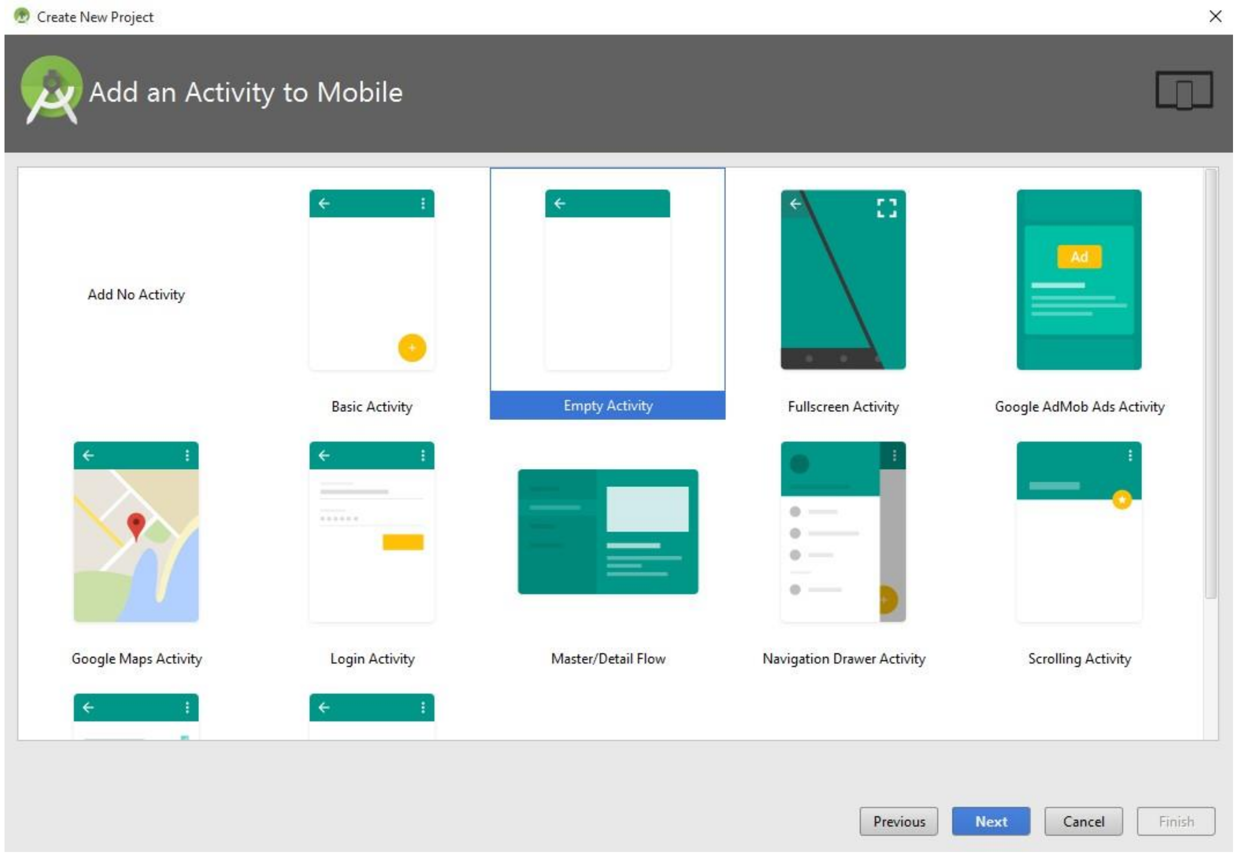
Android Studio
Version 4.1.2

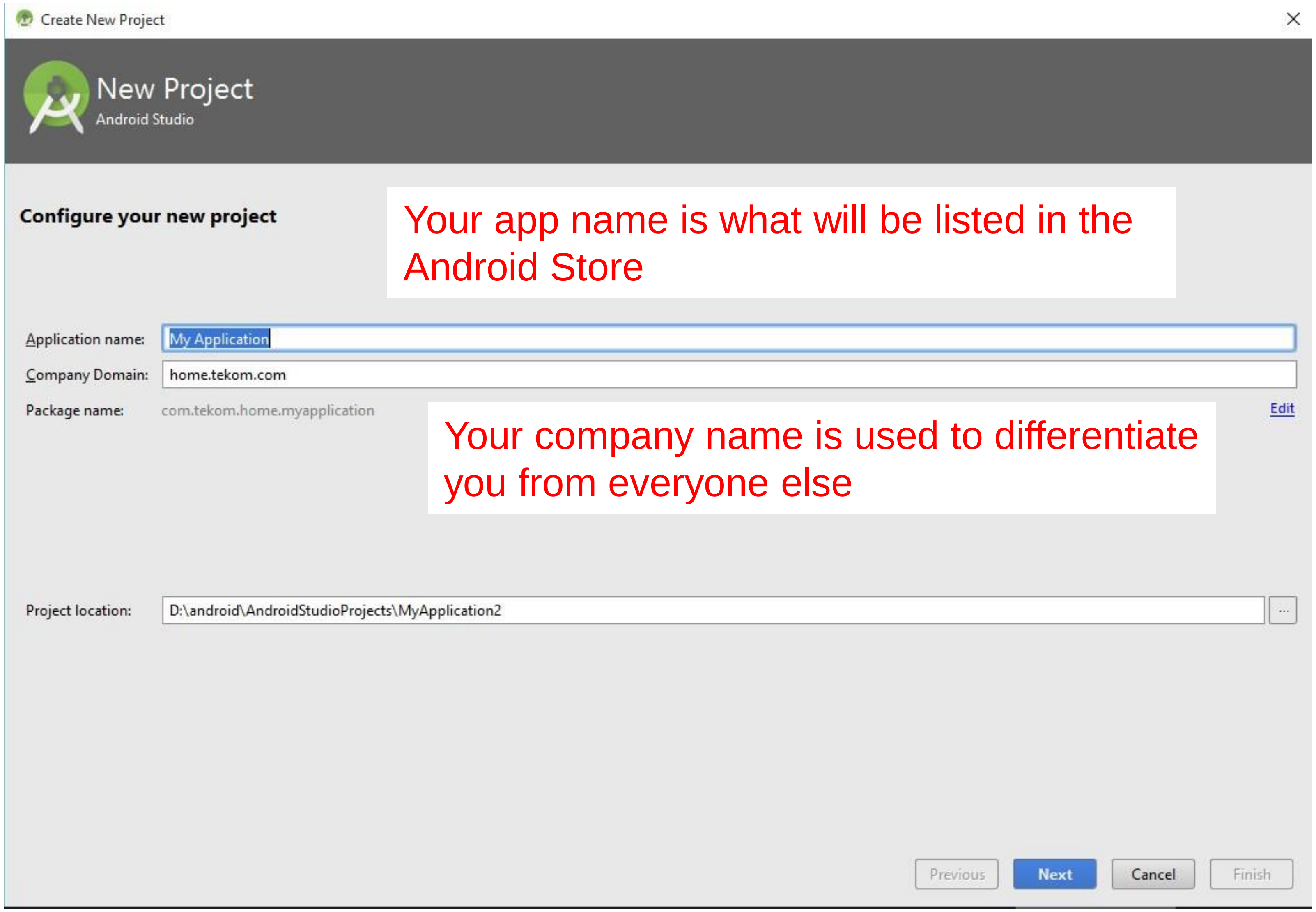
Create New Project
Open an Existing Project
Get from Version Control
Profile or Debug APK
Import Project (Gradle, Eclipse ADT, etc.)
Import an Android Code Sample

Configure
Get Help

NEW VERSION

- Blank – blank activity with an action bar with title and options menu
- Fullscreen – hides the system user interface, action bar shows up on touch
- Login – looks like a login page
- Master/Detail – screen in two sections for menu and details




Create New Project

New Project
Android Studio

Configure your new project

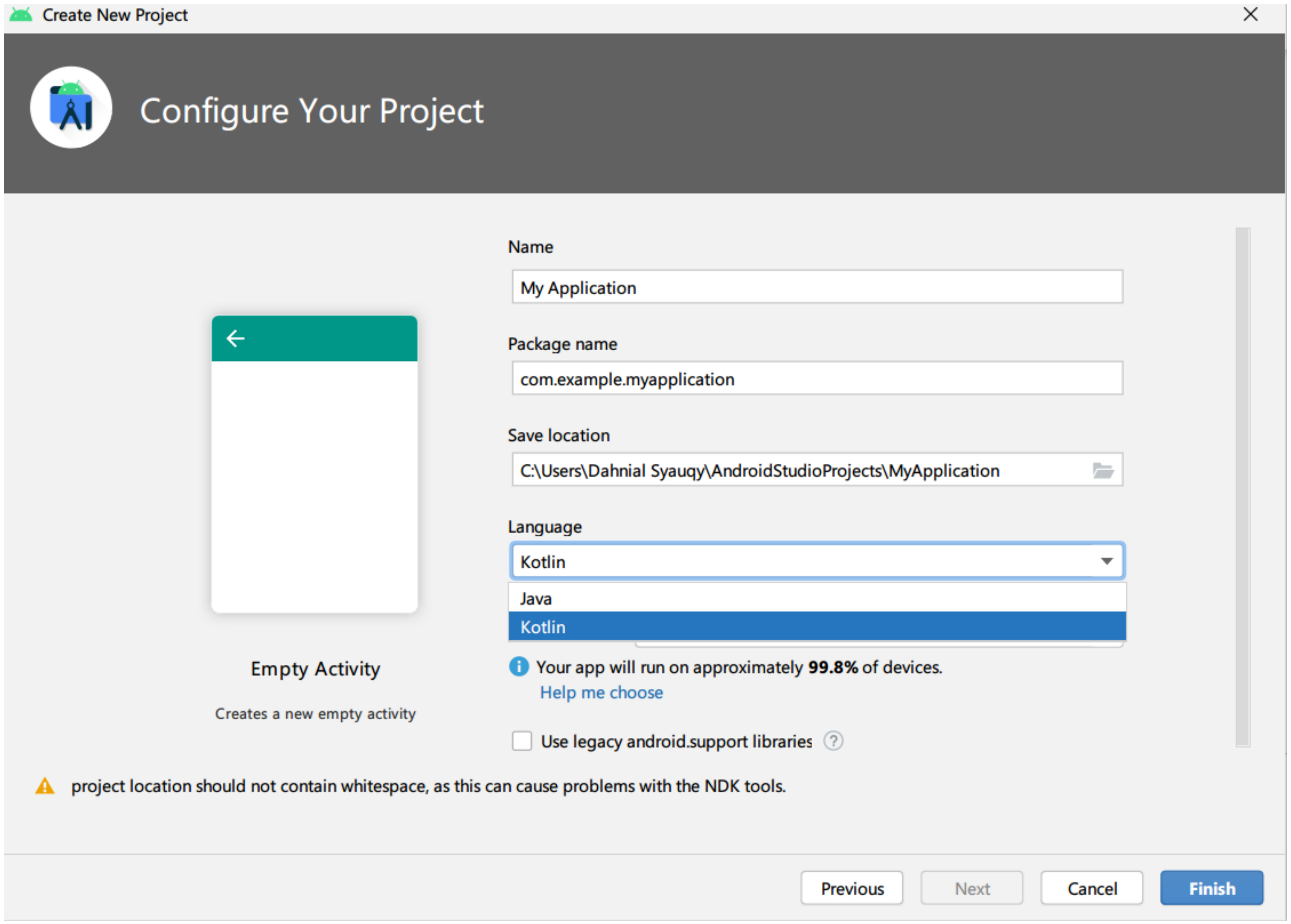
Your app name is what will be listed in the Android Store

Application name: My Application
Company Domain: home.tekom.com
Package name: com.tekom.home.myapplication

Your company name is used to differentiate you from everyone else

Project location: D:\android\AndroidStudioProjects\MyApplication2

Changes: new ver



Create New Project

Configure Your Project

Name: My Application

Package name: com.example.myapplication

Save location: C:\Users\Dahnial Syauqy\AndroidStudioProjects\MyApplication

Language: Kotlin

Java

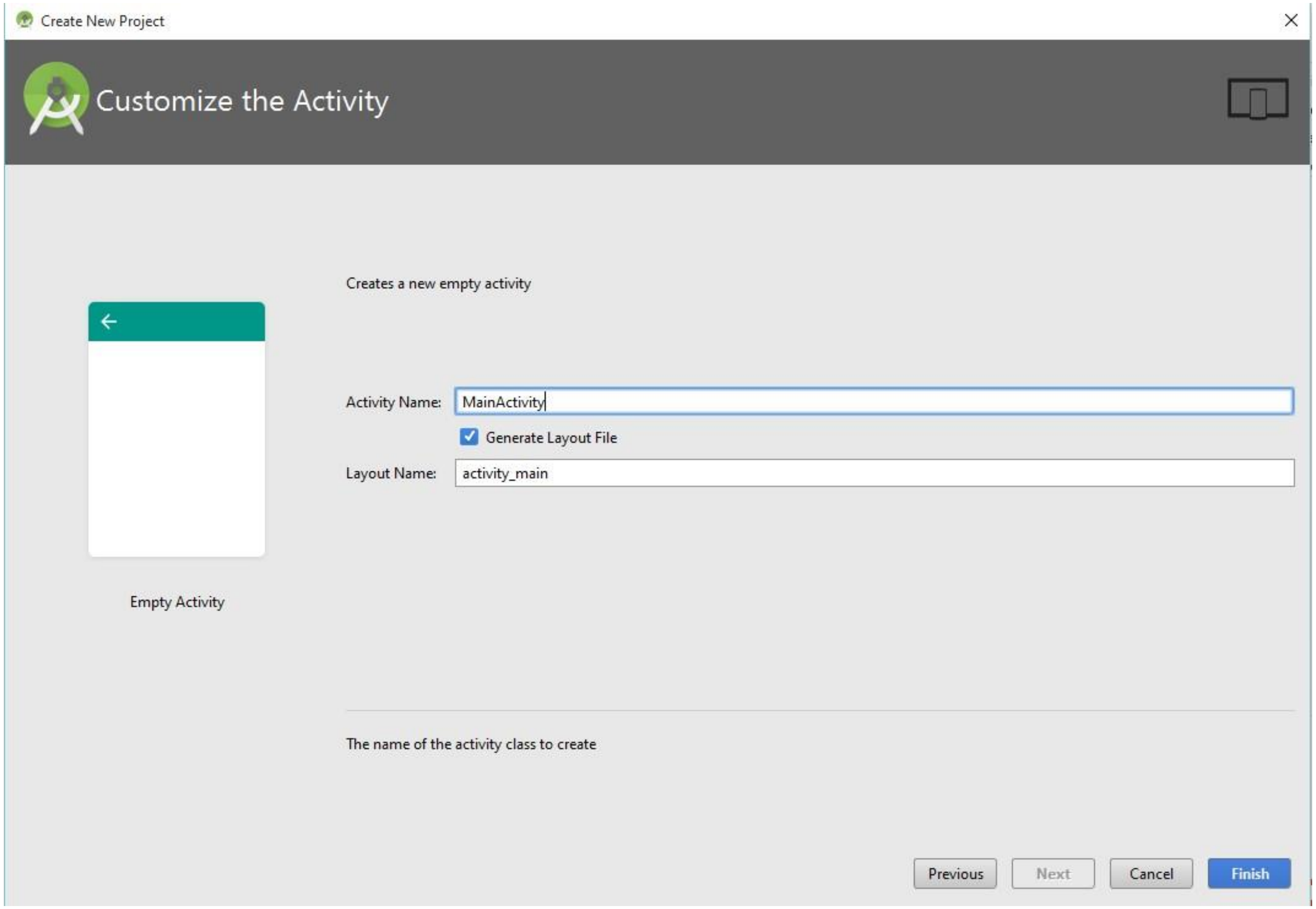
Kotlin

Your app will run on approximately **99.8%** of devices.

Help me choose

☐ Use legacy android.support libraries

project location should not contain whitespace, as this can cause problems with the NDK tools.


Create New Project

Customize the Activity

Creates a new empty activity

Activity Name: MainActivity

[✓] Generate Layout File

Layout Name: activity_main

The name of the activity class to create

What is Activity

An Activity is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map.
Each activity is given a window in which to draw its user interface.
An activity is an instance of the class Activity, part of the Android SDK

- **Activities**
  - Presentation layer for the application you are building
  - For each screen you have, their will be a matching Activity
  - An Activity uses Views to build the user interface
- **Services**
  - Components that run in the background
  - Do not interact with the user
  - Can update your data sources and Activities, and trigger specific notifications

An example

An activity represents a single screen with a user interface.

For example, an email application might have:

- one activity that shows a list of new emails,
- another activity to compose an email, and
- another activity for reading emails.

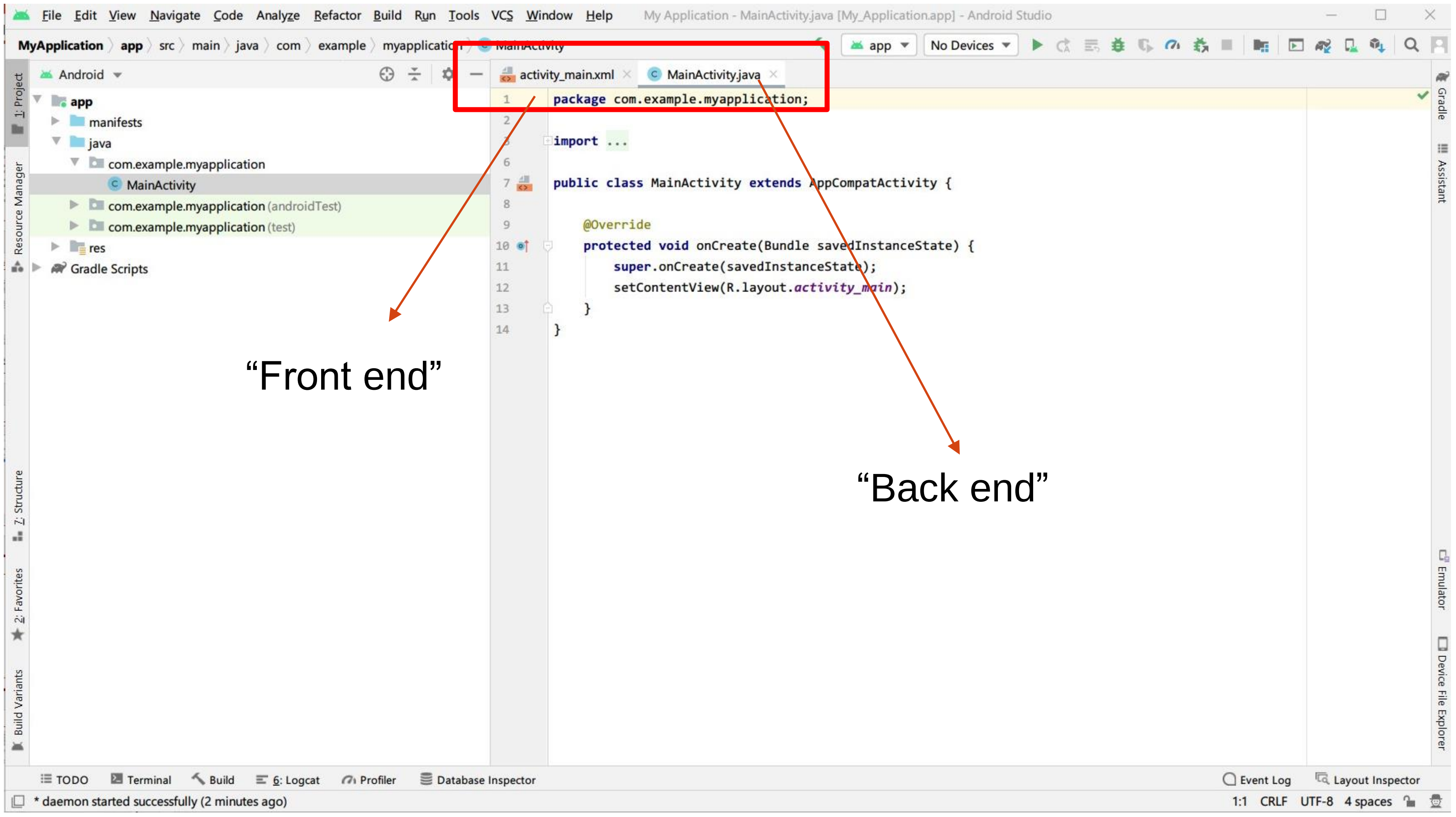
Although the activities work together to form a cohesive user experience in the email application, each one is independent of the others.

As such, a different application can start any one of these activities (if the email application allows it).

For example, a camera application can start the activity in the email application that composes new mail, in order for the user to share a picture.

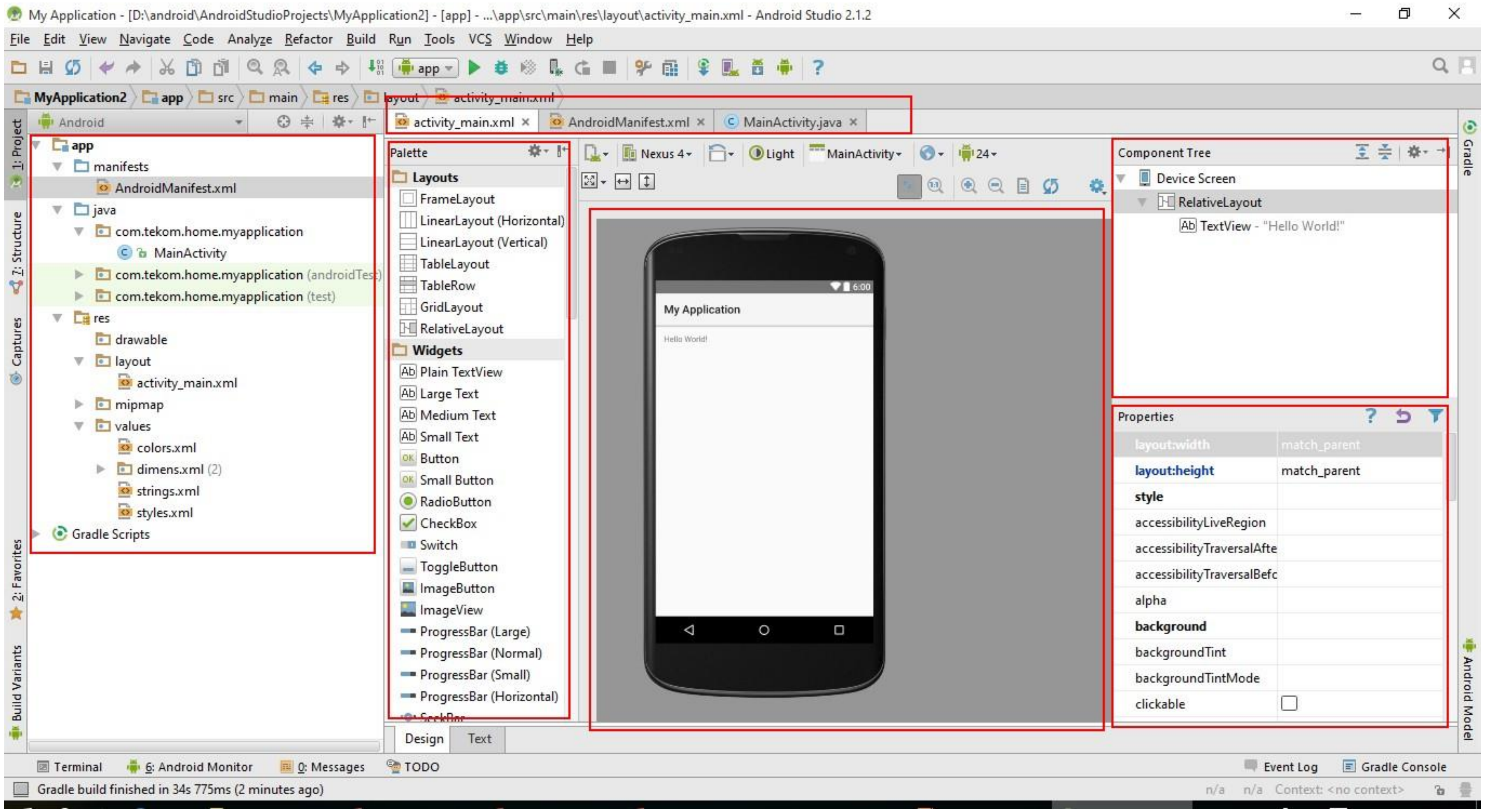
https://developer.android.com/guide/components/fundamentals.html

New version


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
My Application - MainActivity.java [My_Application.app] - Android Studio
MyApplication › app › src › main › java › com › example › myapplication › MainActivity
app
No Devices
Android
app
manifests
java
com.example.myapplication
MainActivity
com.example.myapplication (androidTest)
com.example.myapplication (test)
res
Gradle Scripts
activity_main.xml
MainActivity.java
TODO
Terminal
Build
6: Logcat
Profiler
Database Inspector
Event Log
Layout Inspector
1:1 CRLF UTF-8 4 spaces
“Front end”
“Back end”

```
package com.example.myapplication;

import ...

public class MainActivity extends AppCompatActivity {

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

* daemon started successfully (2 minutes ago)



My Application - [D:\android\AndroidStudioProjects\MyApplication2] - [app] - ...\app\src\main\res\layout\activity_main.xml - Android Studio 2.1.2

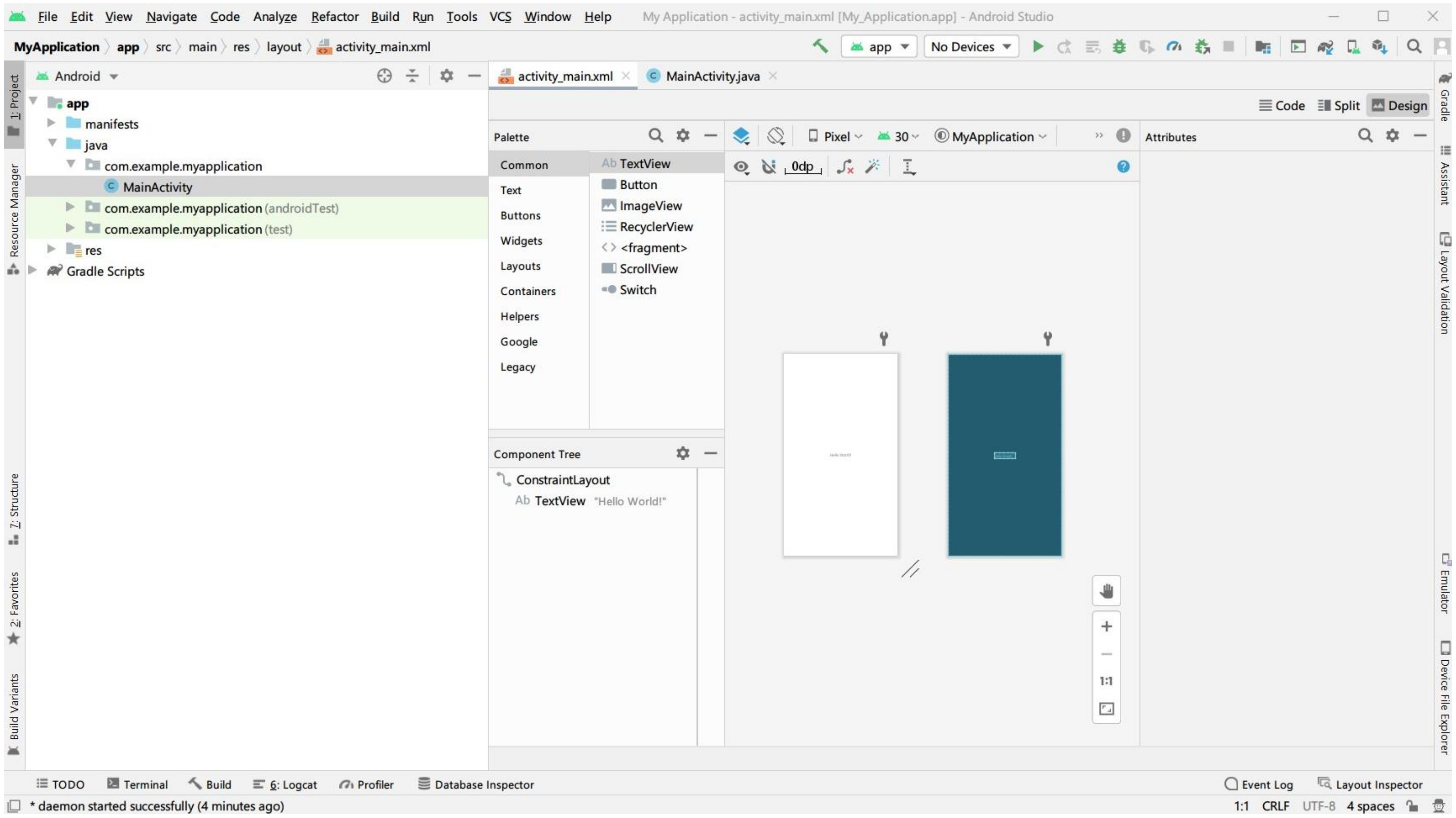
- app
  - manifests
    - AndroidManifest.xml
  - java
    - com.tekom.home.myapplication
      - MainActivity
    - com.tekom.home.myapplication (androidTest)
    - com.tekom.home.myapplication (test)
  - res
    - drawable
    - layout
      - activity_main.xml
    - mipmap
    - values
      - colors.xml
      - dimens.xml (2)
      - strings.xml
      - styles.xml
- Gradle Scripts

layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveRegion	
accessibilityTraversalAfte	
accessibilityTraversalBefc	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	☐

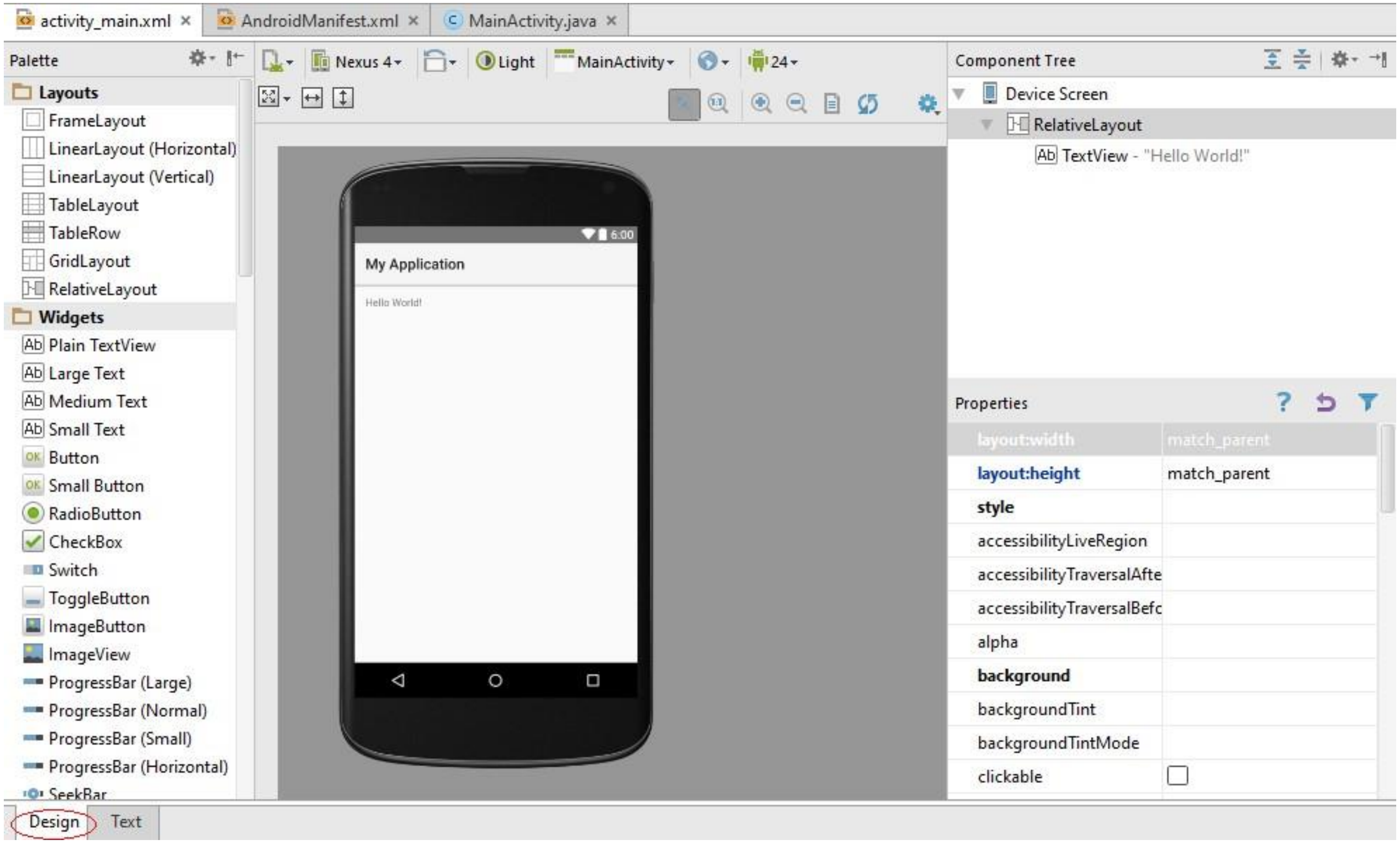
Gradle build finished in 34s 775ms (2 minutes ago)

n/a n/a Context: <no context>

New version

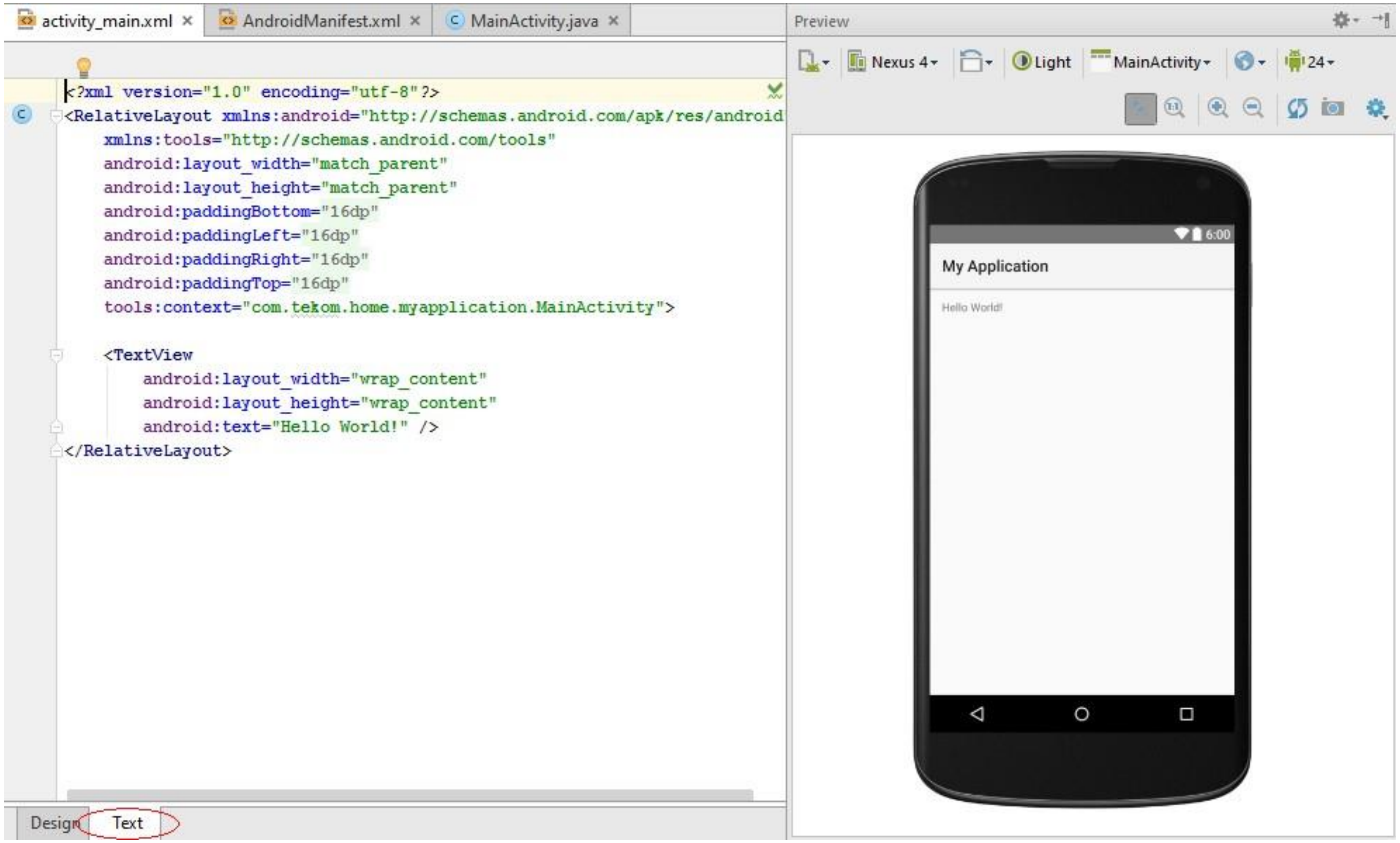

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
My Application - activity_main.xml [My_Application.app] - Android Studio
MyApplication > app > src > main > res > layout > activity_main.xml
app
No Devices
Android
app
manifests
java
com.example.myapplication
MainActivity
com.example.myapplication (androidTest)
com.example.myapplication (test)
res
Gradle Scripts
activity_main.xml
MainActivity.java
Code
Split
Design
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Helpers
Google
Legacy
TextView
Button
ImageView
RecyclerView
<fragment>
ScrollView
Switch
Pixel
30
MyApplication
0dp
Attributes
Component Tree
ConstraintLayout
TextView "Hello World!"
TODO
Terminal
Build
6: Logcat
Profiler
Database Inspector
Event Log
Layout Inspector
* daemon started successfully (4 minutes ago)
1:1 CRLF UTF-8 4 spaces

In “front end” You can work with the layout in “graphical” design mode, or…



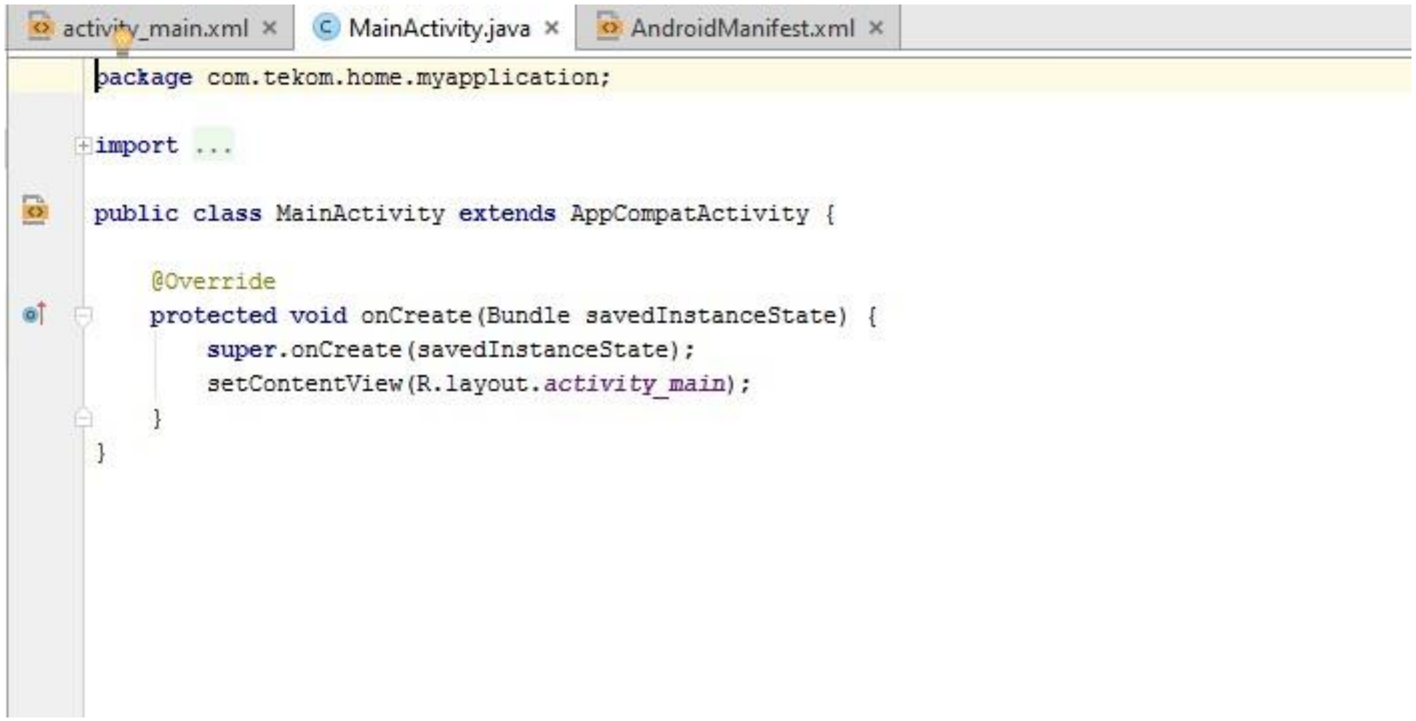
layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveRegion	
accessibilityTraversalAfte	
accessibilityTraversalBefc	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	☐

… in a text mode (based on XML)



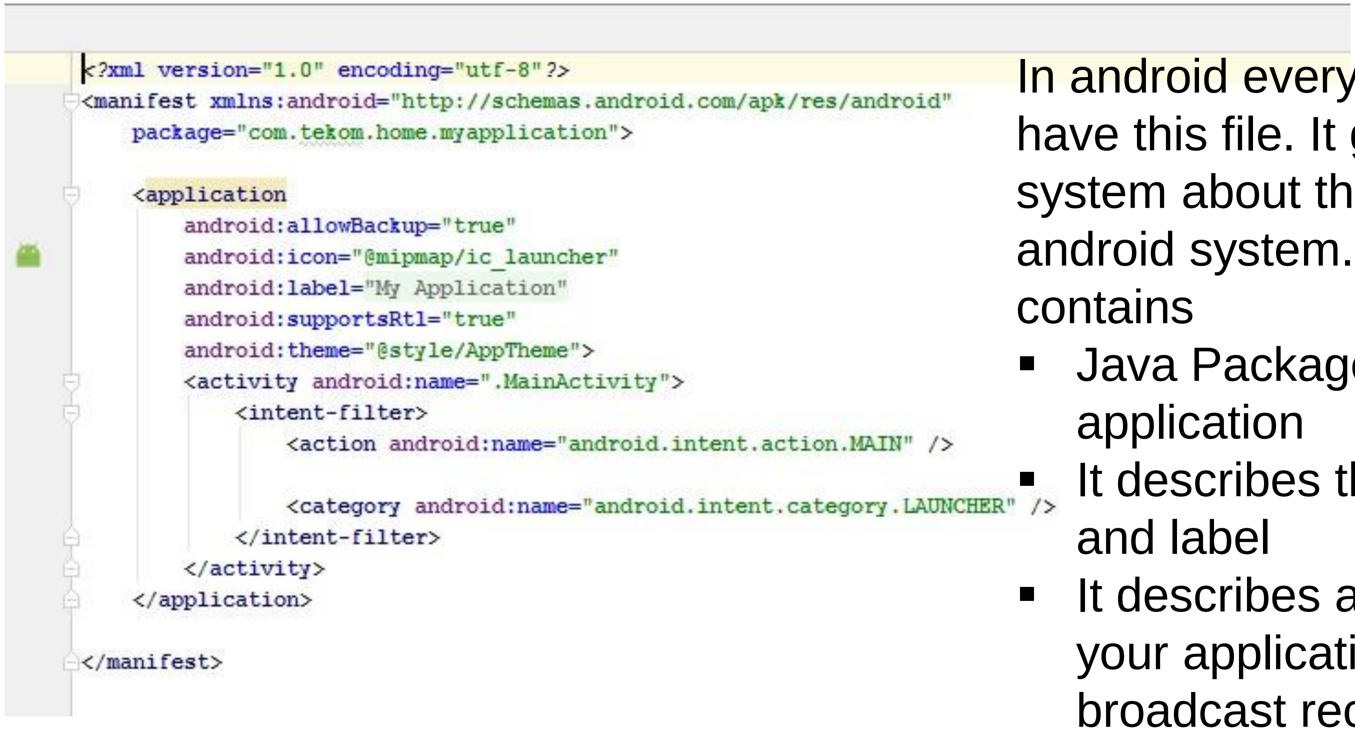
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.tekom.home.myapplication.MainActivity">

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

And this is the java programming “back end”



```
package com.tekom.home.myapplication;

import ...

public class MainActivity extends AppCompatActivity {

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

Android manifest



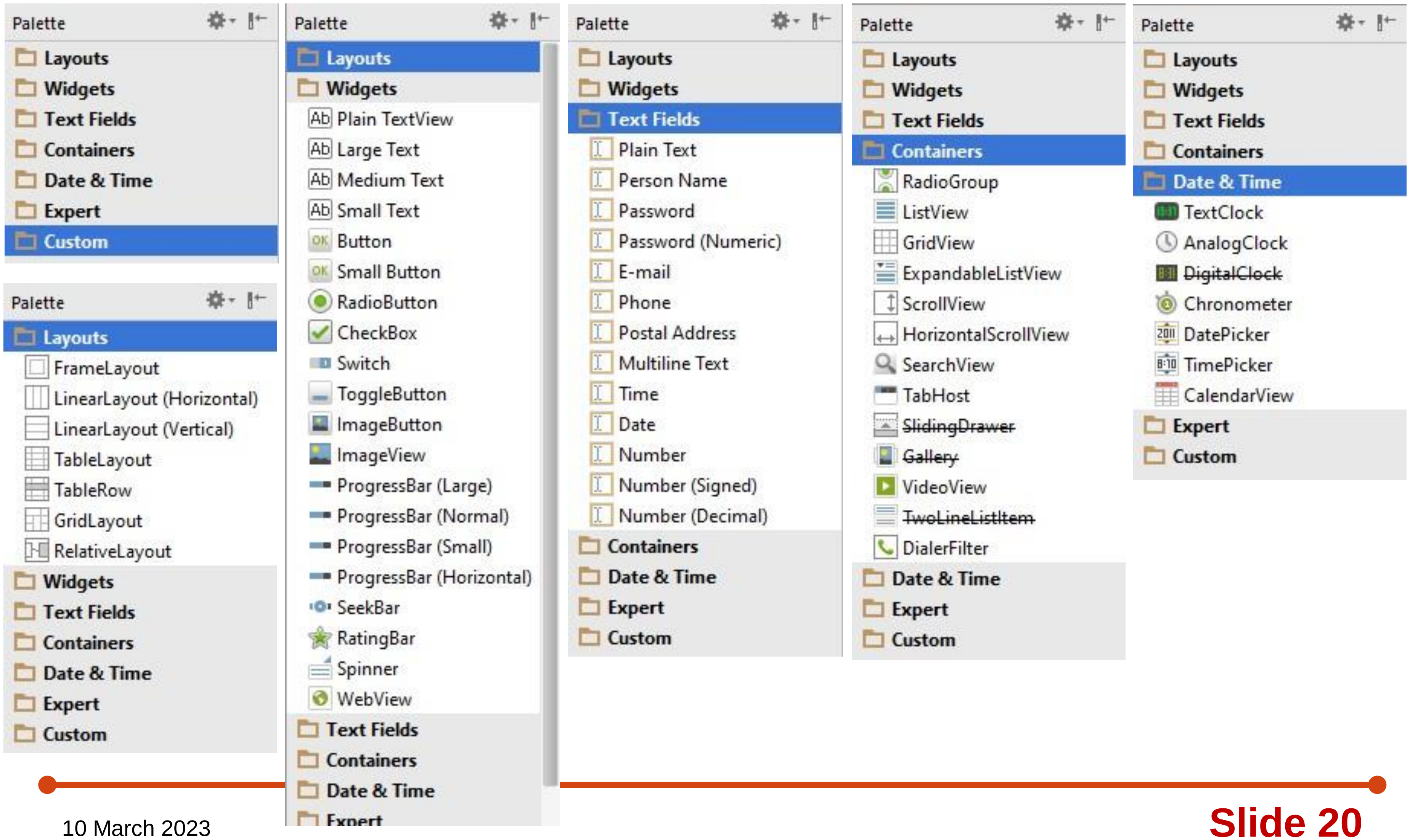
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tekom.home.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="My Application"
        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>
```

AndroidManifest.xml

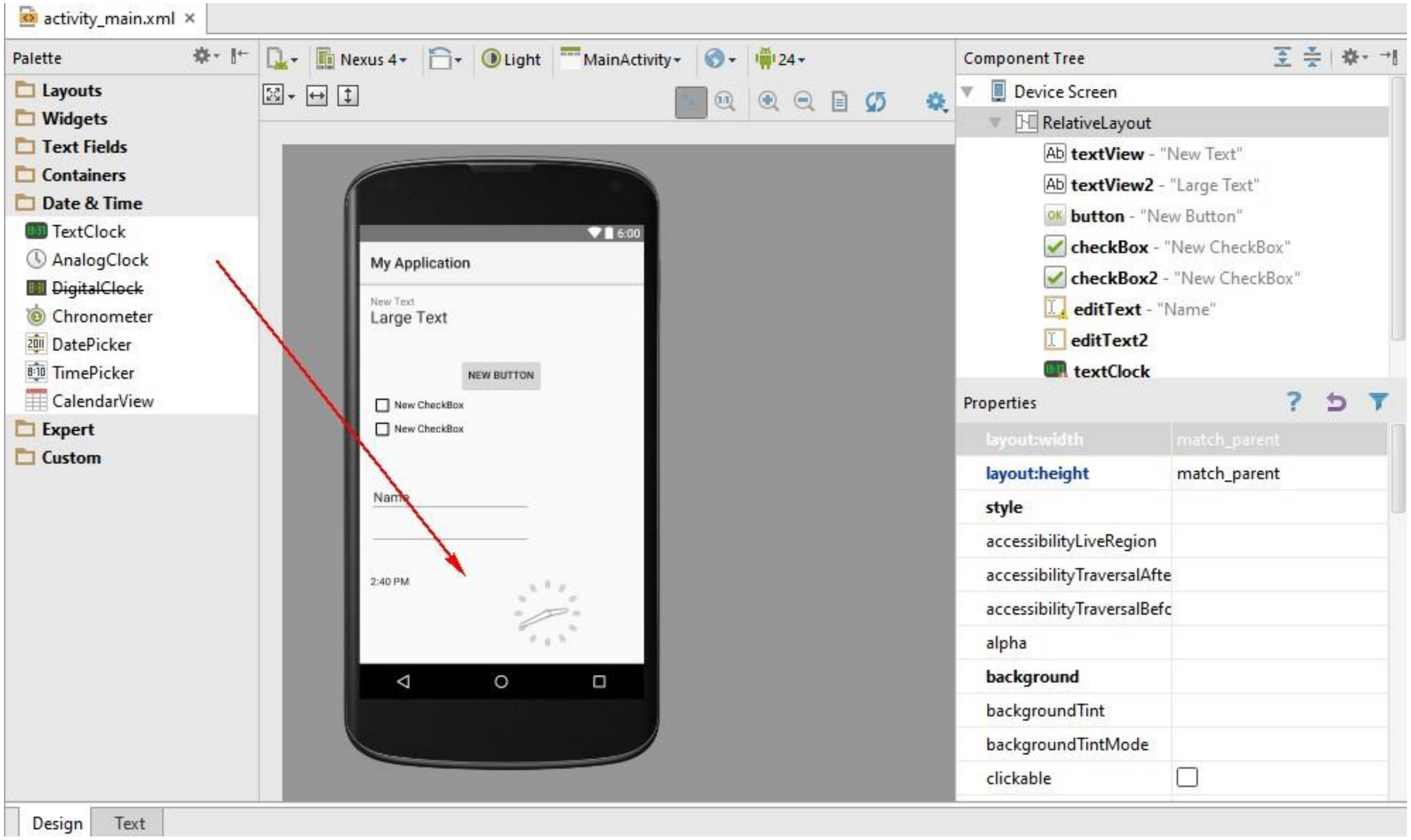
In android every application must have this file. It gives the necessary system about the application to android system. AndroidManifest.xml contains

- Java Package name of the application
- It describes the app icon, theme and label
- It describes all the components of your application like, activities, broadcast receivers, services etc.
- It describes all the permissions your application has to access the restrict part of android system

Try Adding Some widgets instead of only just a “Hello World”!



If you work in graphical design mode, then just drag and drop it (and also adjust its position!)



Component Tree
- Device Screen
  - RelativeLayout
    - textView - "New Text"
    - textView2 - "Large Text"
    - button - "New Button"
    - checkBox - "New CheckBox"
    - checkBox2 - "New CheckBox"
    - editText - "Name"
    - editText2
    - textClock

Properties

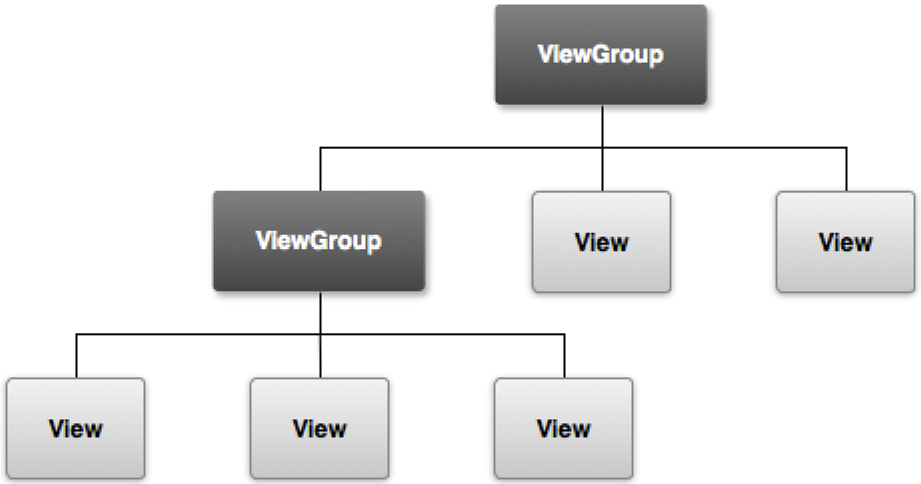
layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveRegion	
accessibilityTraversalAfte	
accessibilityTraversalBefc	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	☐

View and ViewGroup

All user interface elements in an Android app are built using View and ViewGroup objects.

A View is an object that draws something on the screen that the user can interact with.

A ViewGroup is an object that holds other View (and ViewGroup) objects in order to define the layout of the interface.



View and ViewGroup

View

- View objects are the basic building blocks of User Interface(UI) elements in Android.
- View is a simple rectangle box which responds to the user's actions.
- Examples are EditText, Button, CheckBox etc..
- View refers to the android.view.View class, which is the base class of all UI classes.

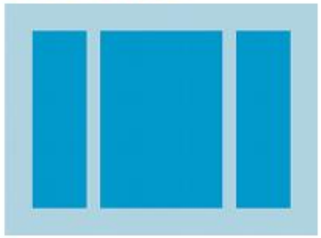
ViewGroup

- ViewGroup is the invisible container. It holds View and ViewGroup
- For example, LinearLayout is the ViewGroup that contains Button(View), and other Layouts also.
- ViewGroup is the base class for Layouts.

Layouts

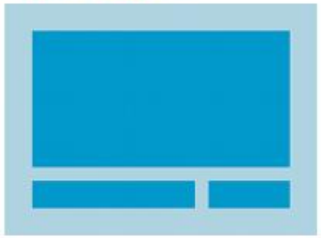
- Specify the position of child views (controls) on the screen
- Common Layout Objects:
  - Linear layout
  - Relative layout
  - Web view
  - List view
  - Grid View

Linear Layout



A layout that organizes its children into a single horizontal or vertical row. It creates a scrollbar if the length of the window exceeds the length of the screen.

Relative Layout

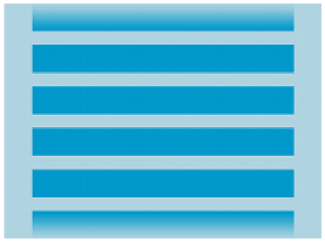


Enables you to specify the location of child objects relative to each other (child A to the left of child B) or to the parent (aligned to the top of the parent).

Web View

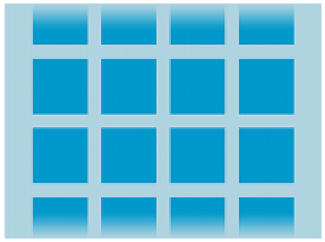
Displays web pages.

List View



Displays a scrolling single column list.

Grid View



Displays a scrolling grid of columns and rows.

In the course of developing Android apps in Android Studio it will be necessary to compile and run an application multiple times.

An Android application may be tested by installing and running it either on a **physical device** or in an ***Android Virtual Device*** *(AVD)* emulator environment.

Before an AVD can be used, it must first be created and configured to match the specification of a particular device model.

Run your first “app”

1. Using AVD------→ then set it up first (if you have the image)
2. Using real device -----→ need to setup USB drivers

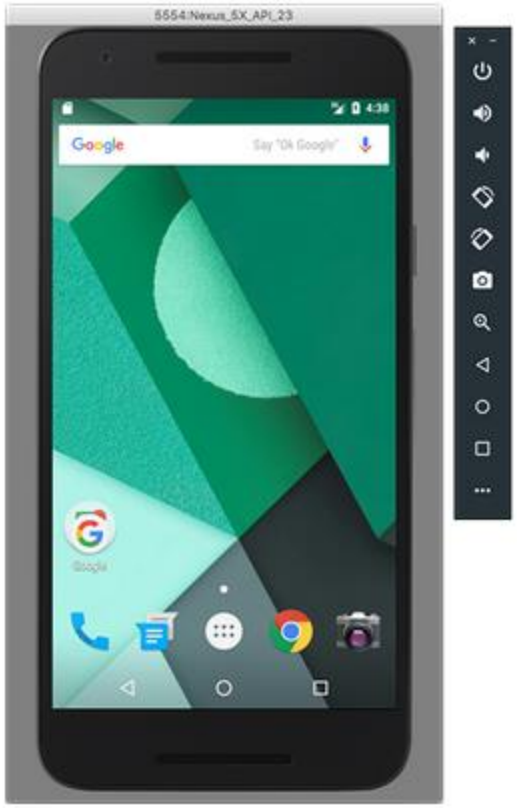
Android Virtual Device

- Development requires either an Android OS device or an emulator

  When building an Android app, it's important that you always test your application on a real device before releasing it to users

- Emulator has limitations:
  - Performance is poor
  - Camera, etc., simulated using computer’s hardware
  - No real phone calls or texts
  - GPS data, battery readings, etc. must be simulated
- Real device is affected by specific hardware and software configuration

Android Virtual Device



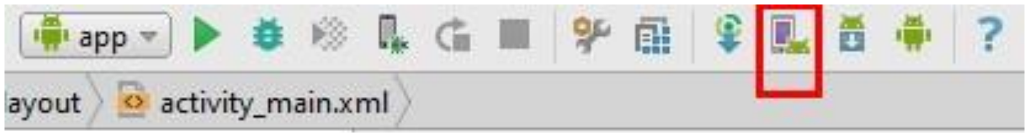
The Android Emulator simulates a device and displays it on your development computer. It lets you prototype, develop, and test Android apps without using a hardware device.

The Android Emulator supports most features of a device, but doesn't include virtual hardware for:
•WiFi
•Bluetooth
•NFC
•SD card insert/eject
•Device-attached headphones
•USB

Android Virtual Device (AVD)

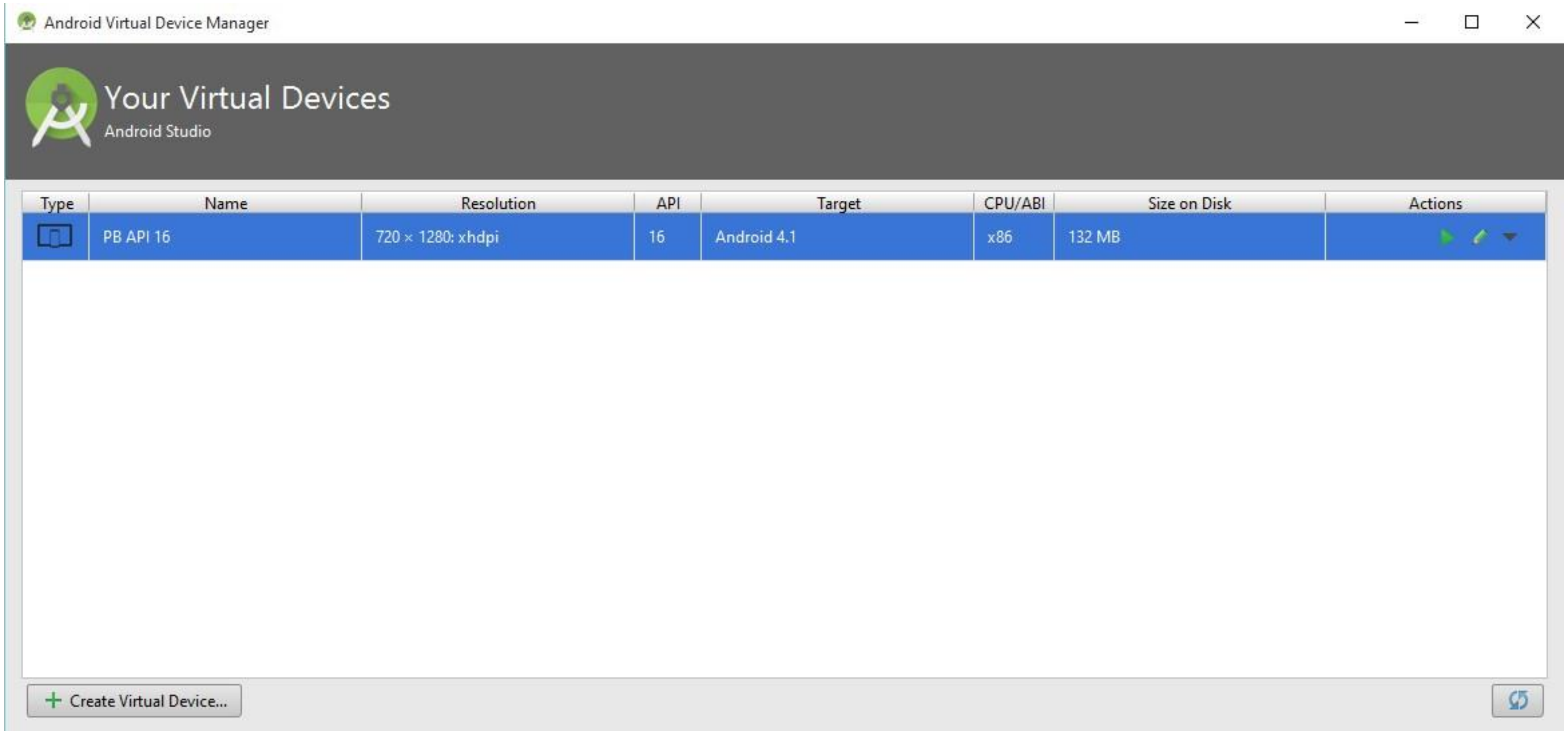
AVDs are essentially emulators that allow Android applications to be tested without the necessity to install the application on a physical Android based device.

An AVD may be configured to emulate a variety of hardware features including options such as screen size, memory capacity and the presence or otherwise of features such as a camera, GPS navigation support or an accelerometer.



AVD manager

We will create an AVD

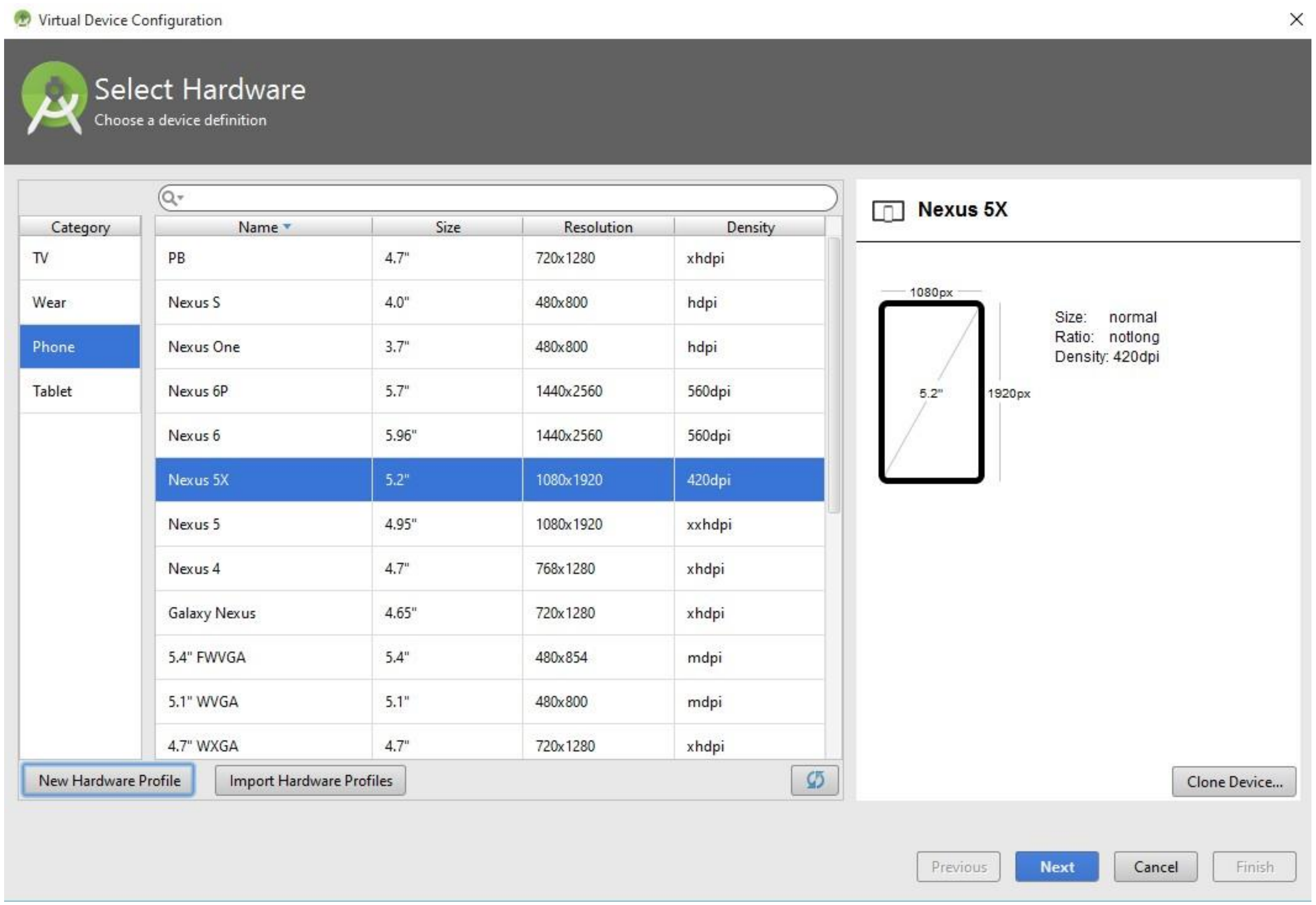


Android Virtual Device Manager

Your Virtual Devices
Android Studio

Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	PB API 16	720 × 1280: xhdpi	16	Android 4.1	x86	132 MB	

You can choose the existing device profile, or **create a new one**



Virtual Device Configuration

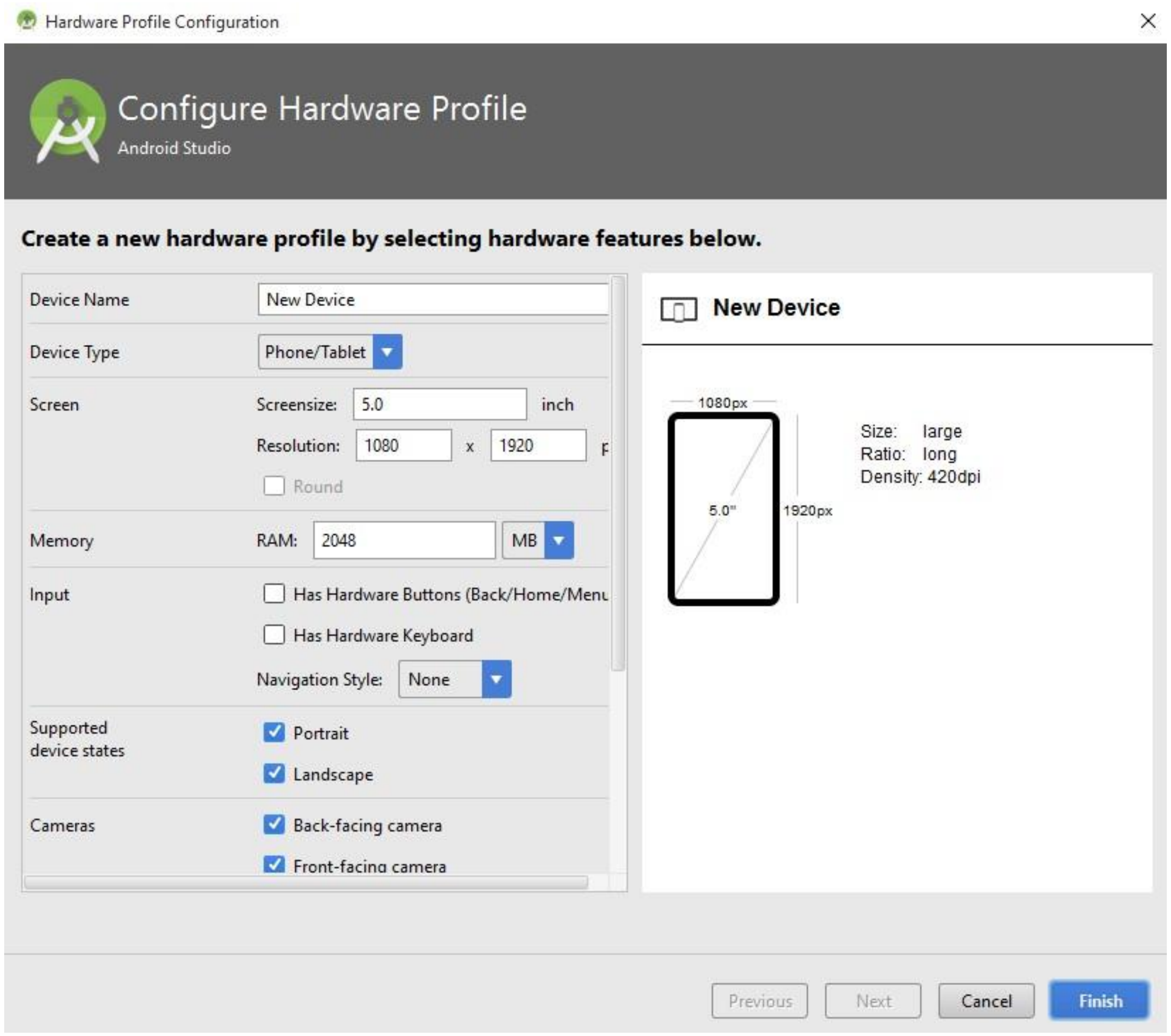
Select Hardware

Choose a device definition

Category	Name	Size	Resolution	Density
TV	PB	4.7"	720x1280	xhdpi
Wear	Nexus S	4.0"	480x800	hdpi
Phone	Nexus One	3.7"	480x800	hdpi
Tablet	Nexus 6P	5.7"	1440x2560	560dpi
	Nexus 6	5.96"	1440x2560	560dpi
	Nexus 5X	5.2"	1080x1920	420dpi
	Nexus 5	4.95"	1080x1920	xxhdpi
	Nexus 4	4.7"	768x1280	xhdpi
	Galaxy Nexus	4.65"	720x1280	xhdpi
	5.4" FWVGA	5.4"	480x854	mdpi
	5.1" WVGA	5.1"	480x800	mdpi
	4.7" WXGA	4.7"	720x1280	xhdpi

Nexus 5X

Size: normal
Ratio: notlong
Density: 420dpi



Hardware Profile Configuration

Configure Hardware Profile
Android Studio

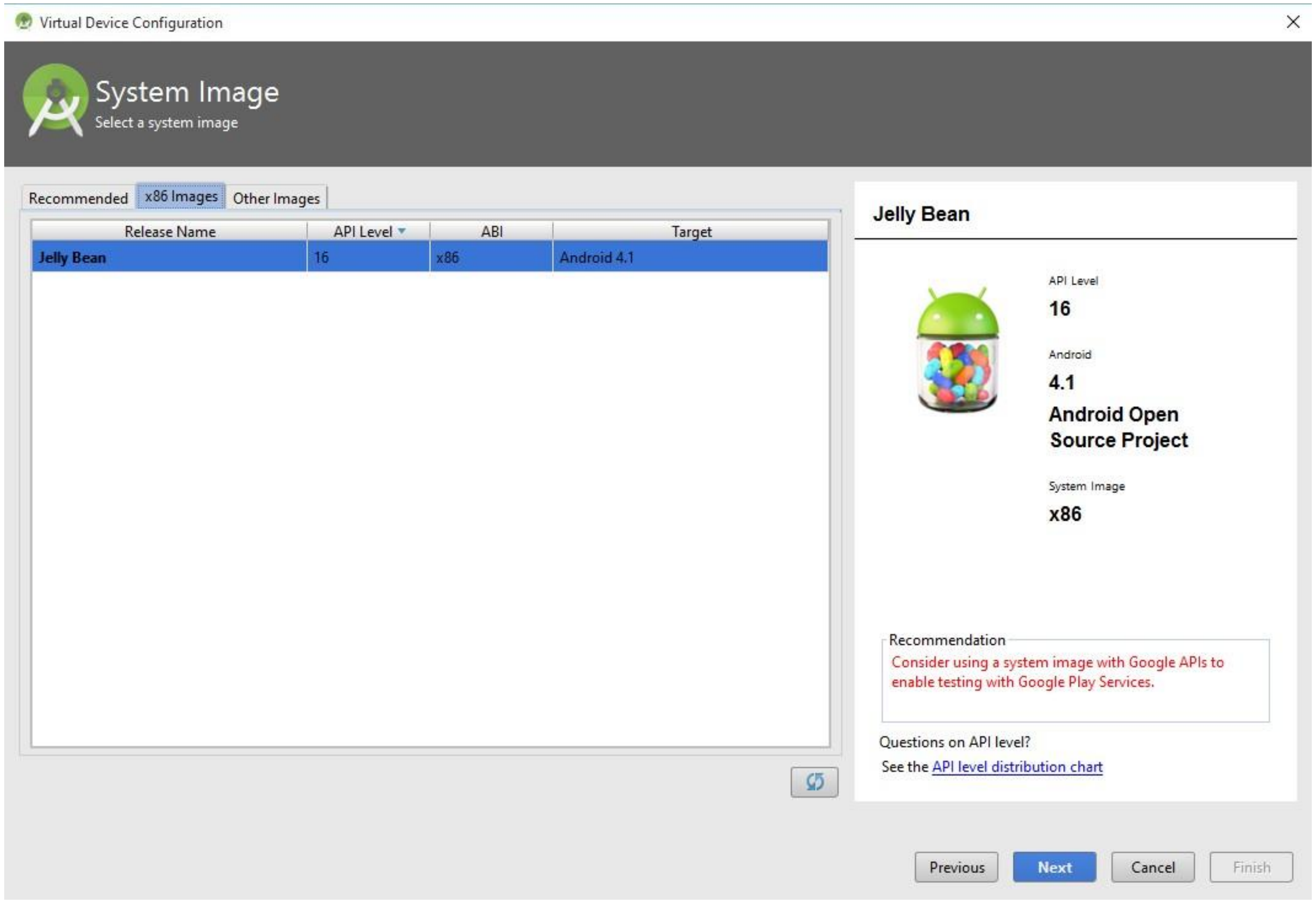
Create a new hardware profile by selecting hardware features below.

Device Name	New Device
Device Type	Phone/Tablet
Screen	Screensize: 5.0 inch Resolution: 1080 x 1920 p ☐ Round
Memory	RAM: 2048 MB
Input	☐ Has Hardware Buttons (Back/Home/Menu ☐ Has Hardware Keyboard Navigation Style: None
Supported device states	☑ Portrait ☑ Landscape
Cameras	☑ Back-facing camera ☑ Front-facing camera

New Device

Size: large
Ratio: long
Density: 420dpi

An AVD need an image, we have 16-based API level image, and also 23-API based one



Release Name	API Level	ABI	Target
Jelly Bean	16	x86	Android 4.1

Jelly Bean

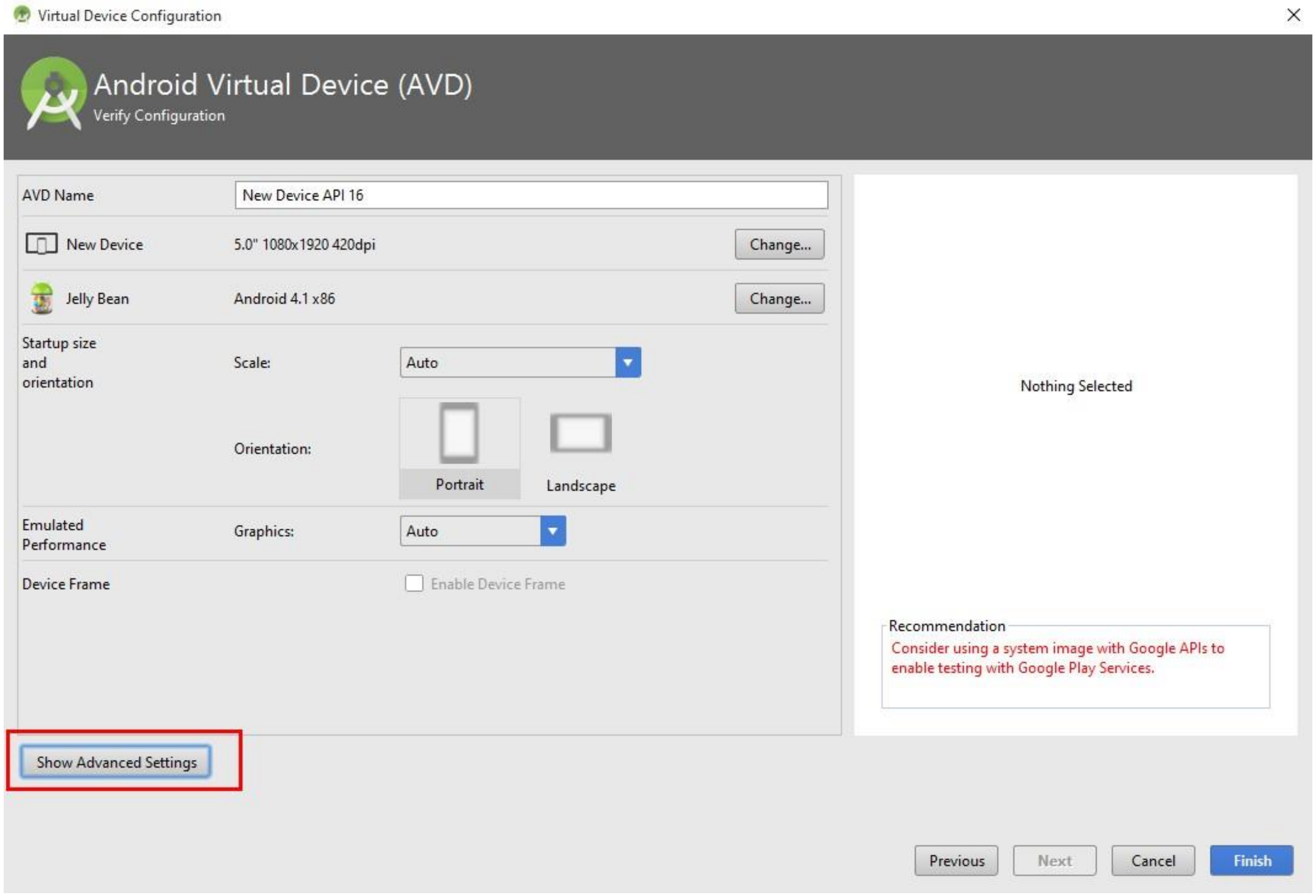
API Level
16

Android
4.1
Android Open Source Project

System Image
x86

Recommendation
Consider using a system image with Google APIs to enable testing with Google Play Services.

Questions on API level?
See the API level distribution chart



Virtual Device Configuration

Android Virtual Device (AVD)

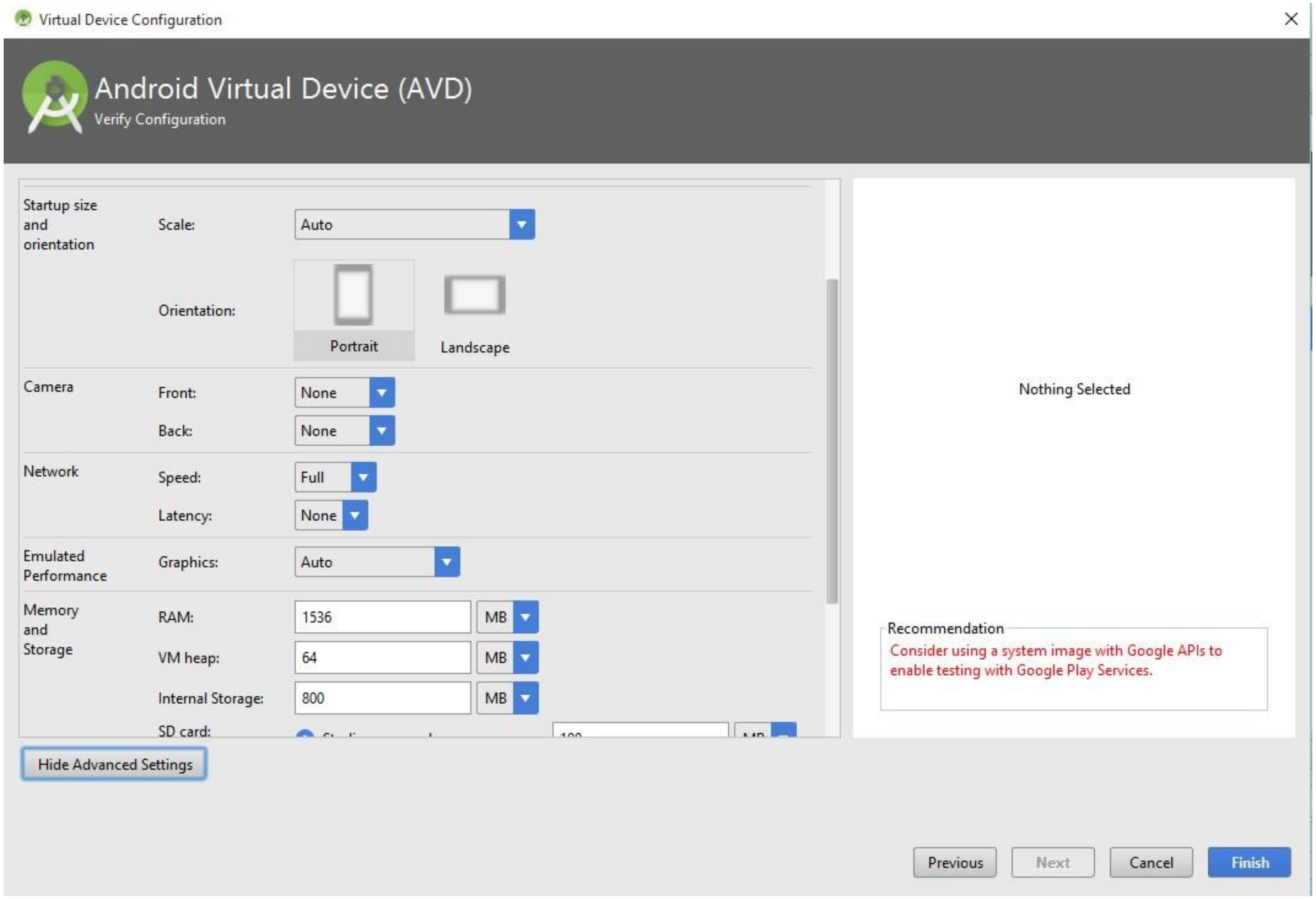
Verify Configuration

AVD Name	New Device API 16	
New Device	5.0" 1080x1920 420dpi	Change...
Jelly Bean	Android 4.1 x86	Change...
Startup size and orientation	Scale: Auto	
	Orientation: [✓ Portrait] Landscape	
Emulated Performance	Graphics: Auto	
Device Frame	☐ Enable Device Frame	

Nothing Selected

Recommendation

Consider using a system image with Google APIs to enable testing with Google Play Services.



Virtual Device Configuration

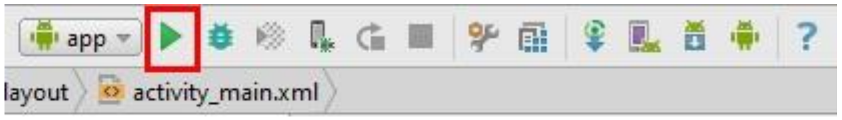
Android Virtual Device (AVD)
Verify Configuration

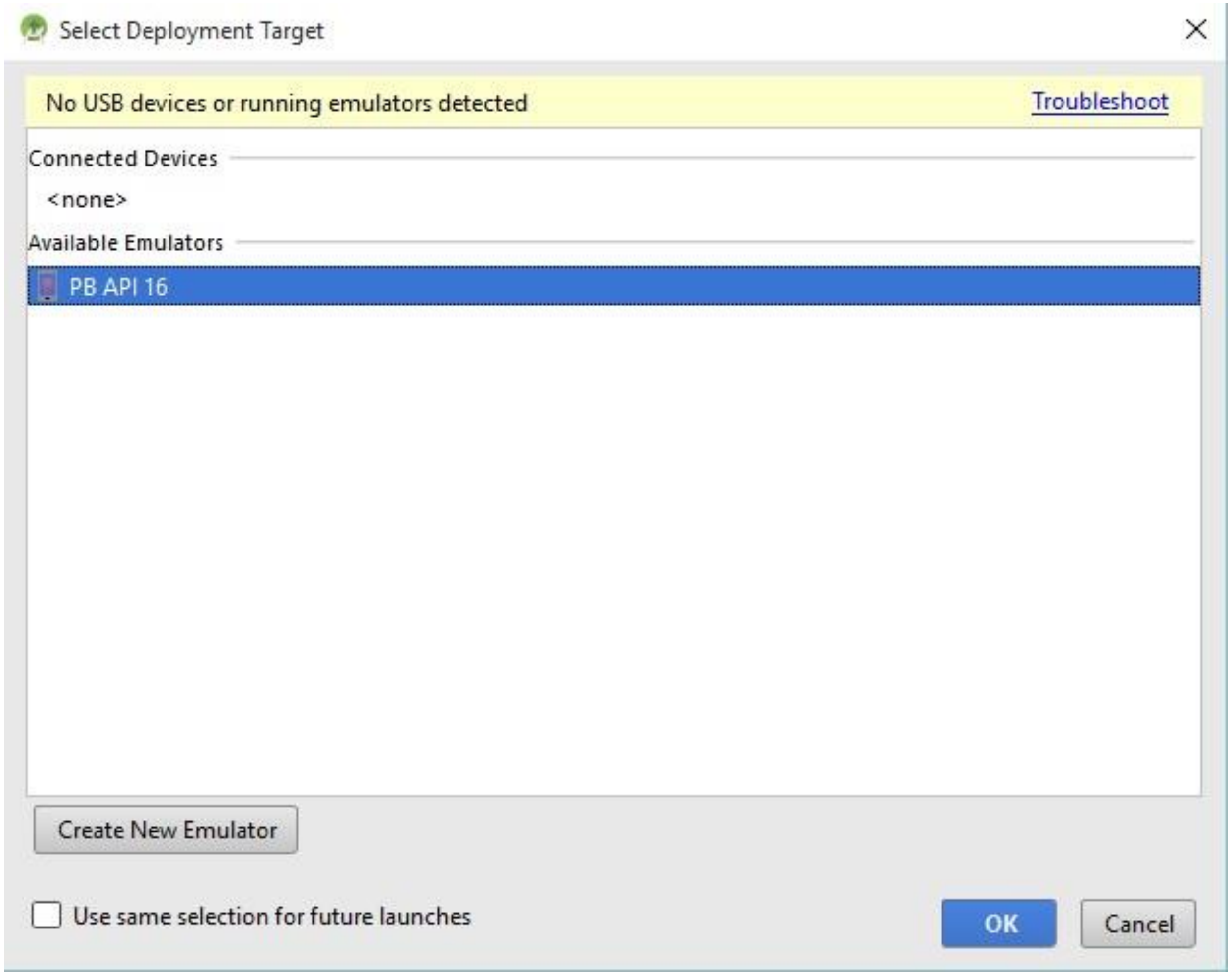
Startup size and orientation	Scale:	Auto
	Orientation:	[✓ Portrait] Landscape
Camera	Front:	None
	Back:	None
Network	Speed:	Full
	Latency:	None
Emulated Performance	Graphics:	Auto
Memory and Storage	RAM:	1536 MB
	VM heap:	64 MB
	Internal Storage:	800 MB
	SD card:	

Nothing Selected

Recommendation
Consider using a system image with Google APIs to enable testing with Google Play Services.

Run your app




Select Deployment Target

No USB devices or running emulators detected

Troubleshoot

Connected Devices

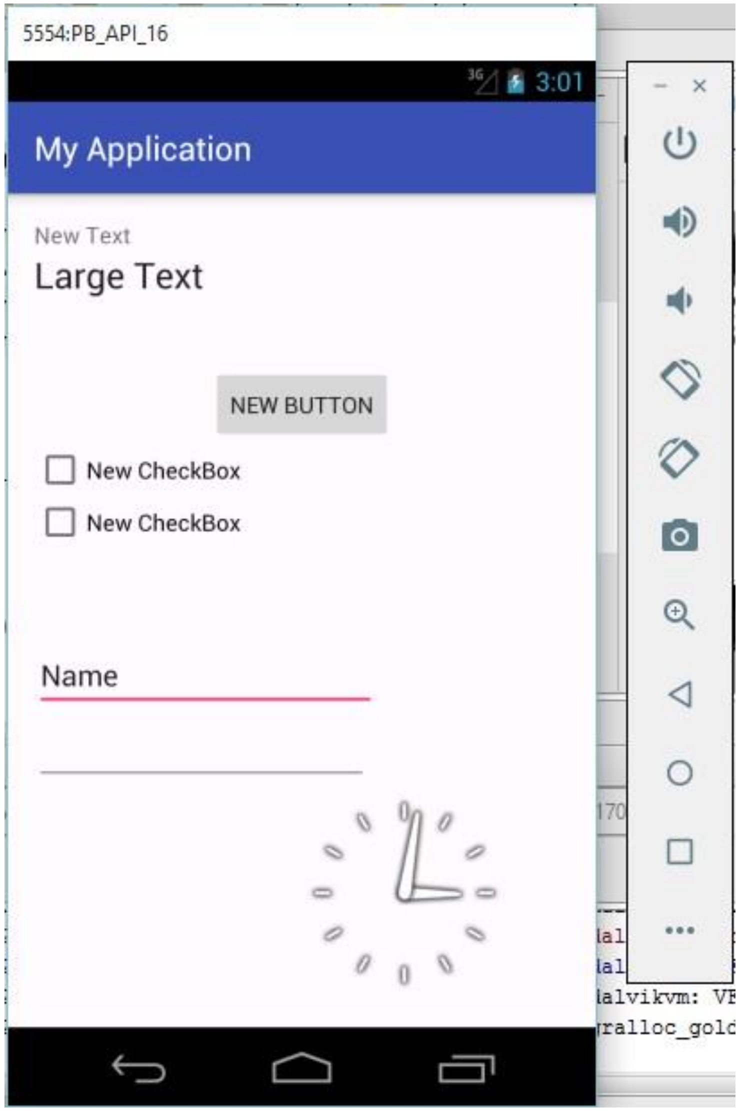
<none>

Available Emulators

PB API 16

What is Gradle? The basic answer is: Gradle is a Build System.

The result …


5554:PB_API_16
3:01
My Application
New Text
Large Text
NEW BUTTON
New CheckBox
New CheckBox
Name

Whilst much can be achieved by testing applications using an Android Virtual Device (AVD), there is no substitute for **performing real world application testing** on a physical Android device and there are a number of **Android features that are only available** on physical Android devices.

Communication with both AVD instances and connected Android devices is handled by the *Android Debug Bridge (ADB)*

Running your App on a Device

1. You need an android device at the min SDK or higher
2. You need a USB cable of appropriate type
3. You need to unlock developer mode on the

device

4. You might want to setup Paths
5. You may need to install USB drivers

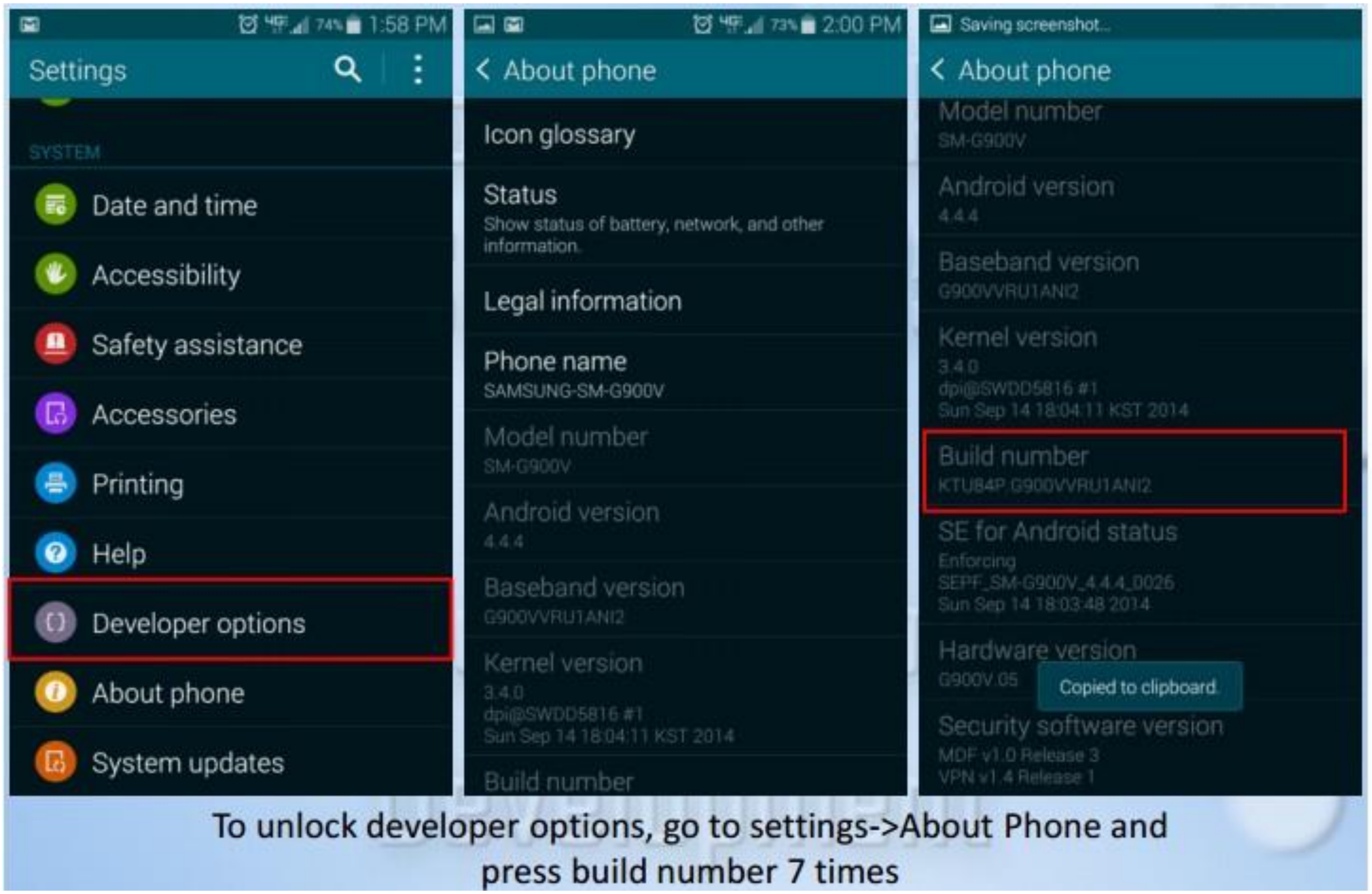
Why an ADB

The primary purpose of the ADB is to facilitate interaction between a development system, in this case Android Studio, and both AVD emulators and physical Android devices for the purposes of running and debugging applications.

The ADB consists of a client, a server process running in the background on the development system and a daemon background process running in either AVDs or real Android devices such as phones and tablets.

The ADB client can take a variety of forms. For example, a client is provided in the form of a command-line tool named *adb* located in the Android SDK *platform-tools* subdirectory. Similarly, Android Studio also has a built-in client.

ADB setup


Settings

SYSTEM

Date and time

Accessibility

Safety assistance

Accessories

Printing

Help

Developer options

About phone

System updates

About phone

Icon glossary

Status
Show status of battery, network, and other information.

Legal information

Phone name
SAMSUNG-SM-G900V

Model number
SM-G900V

Android version
4.4.4

Baseband version
G900VVRU1ANI2

Kernel version
3.4.0
dpi@SWDD5816 #1
Sun Sep 14 18:04:11 KST 2014

Build number

About phone

Model number
SM-G900V

Android version
4.4.4

Baseband version
G900VVRU1ANI2

Kernel version
3.4.0
dpi@SWDD5816 #1
Sun Sep 14 18:04:11 KST 2014

Build number
KTU84P.G900VVRU1ANI2

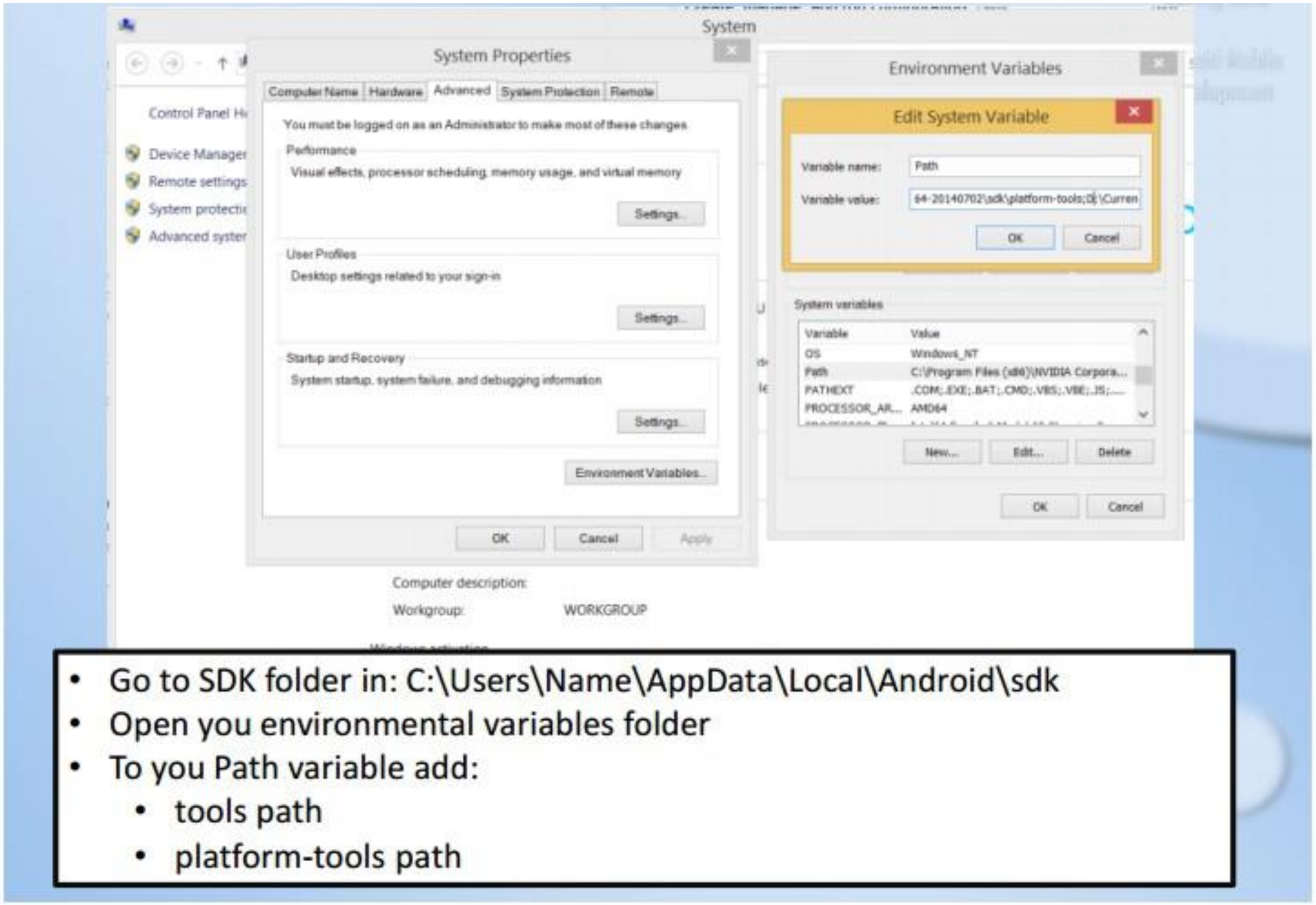
SE for Android status
Enforcing
SEPF_SM-G900V_4.4.4_0026
Sun Sep 14 18:03:48 2014

Hardware version
G900V.05

Copied to clipboard.

Security software version
MDF v1.0 Release 3
VPN v1.4 Release 1

To unlock developer options, go to settings->About Phone and press build number 7 times



You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles
Desktop settings related to your sign-in

Startup and Recovery
System startup, system failure, and debugging information

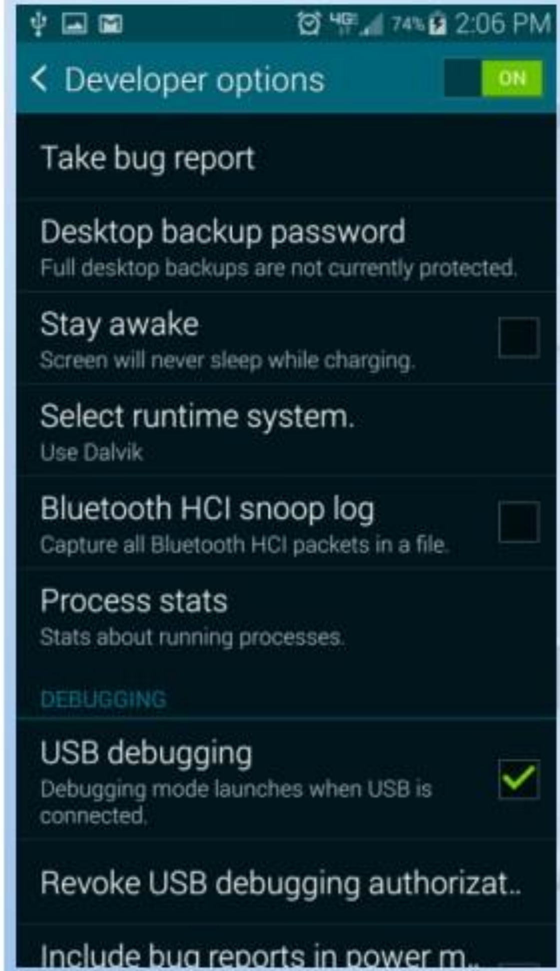
Variable name: Path
Variable value: 64-20140702\sdk\platform-tools;C:\Curren

System variables

Variable	Value
OS	Windows_NT
Path	C:\Program Files (x86)\NVIDIA Corpora...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;...
PROCESSOR_AR...	AMD64

Computer description:
Workgroup: WORKGROUP

- Go to SDK folder in: C:\Users\Name\AppData\Local\Android\sdk
- Open you environmental variables folder
- To you Path variable add:
  - tools path
  - platform-tools path


2:06 PM

Developer options ON

Take bug report

Desktop backup password
Full desktop backups are not currently protected.

Stay awake ☐
Screen will never sleep while charging.

Select runtime system.
Use Dalvik

Bluetooth HCI snoop log ☐
Capture all Bluetooth HCI packets in a file.

Process stats
Stats about running processes.

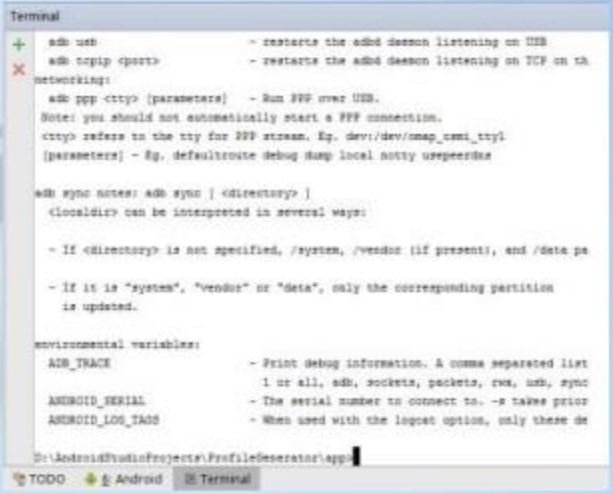
DEBUGGING

USB debugging ✓
Debugging mode launches when USB is connected.

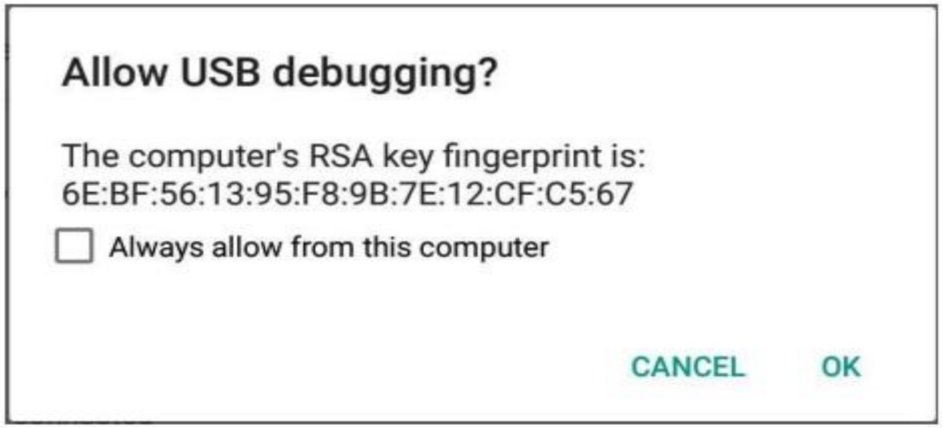
Revoke USB debugging authorizat..

Include bug reports in power m..

- Under **Applications, check Unknown Sources**
- Enable USB debugging
- Connect to your computer (USB)
- Launch your command prompt
- Start->Run cmd or the terminal in studio
- Type the following:
  adb –devices
- You should see the device now!


Terminal

```
  adb usb                      - restarts the adbd daemon listening on USB
  adb tcpip <port>             - restarts the adbd daemon listening on TCP on th
networking:
  adb ppp <tty> [parameters]   - Run PPP over USB.
 Note: you should not automatically start a PPP connection.
 <tty> refers to the tty for PPP stream. Eg. dev:/dev/omap_csmi_tty1
 [parameters] - Eg. defaultroute debug dump local notty usepeerdns

adb sync notes: adb sync [ <directory> ]
  <localdir> can be interpreted in several ways:

  - If <directory> is not specified, /system, /vendor (if present), and /data pa

  - If it is "system", "vendor" or "data", only the corresponding partition
    is updated.

environmental variables:
  ADB_TRACE                    - Print debug information. A comma separated list
                                 1 or all, adb, sockets, packets, rwx, usb, sync
  ANDROID_SERIAL               - The serial number to connect to. -s takes prior
  ANDROID_LOG_TAGS             - When used with the logcat option, only these de

D:\AndroidStudioProjects\ProfileGenerator\app>
```


Allow USB debugging?

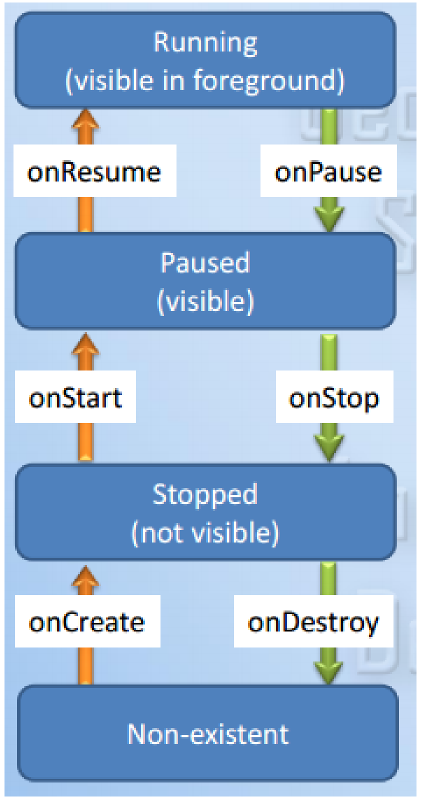
The computer's RSA key fingerprint is:
6E:BF:56:13:95:F8:9B:7E:12:CF:C5:67

☐ Always allow from this computer

Android Activity Lifecycle

This diagram shows the different states your activities can be in

When states are changed, our activity has methods called automatically by the Activity Manager



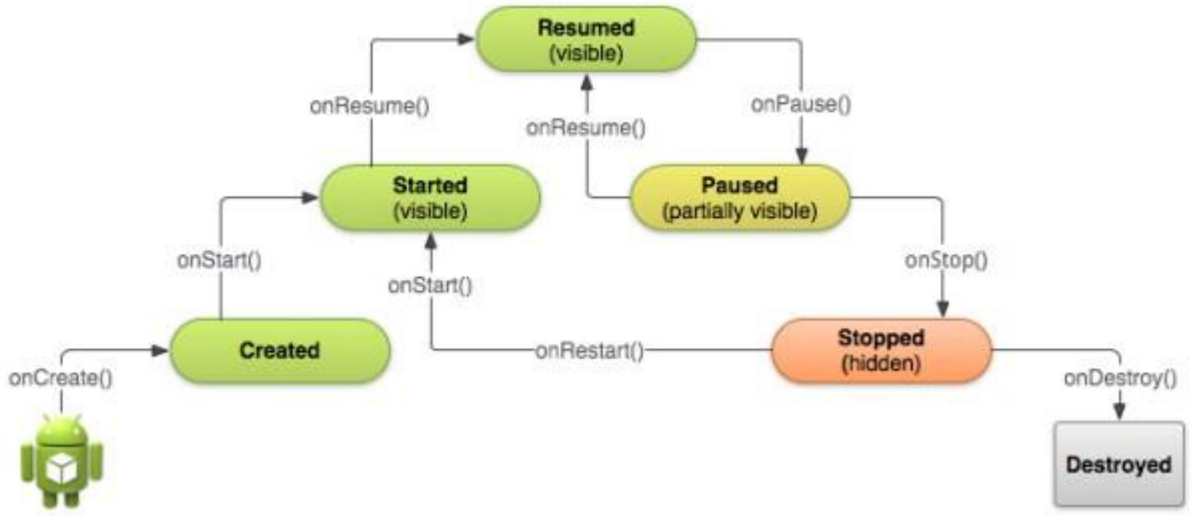
Active / Running – The activity is at the top of the Activity Stack, is the foreground task visible on the device screen, has focus and is currently interacting with the user.

Paused – The activity is visible to the user but does not currently have focus (typically because this activity is partially obscured by the current *active* activity).

Stopped – The activity is currently not visible to the user (in other words it is totally obscured on the device display by other activities).

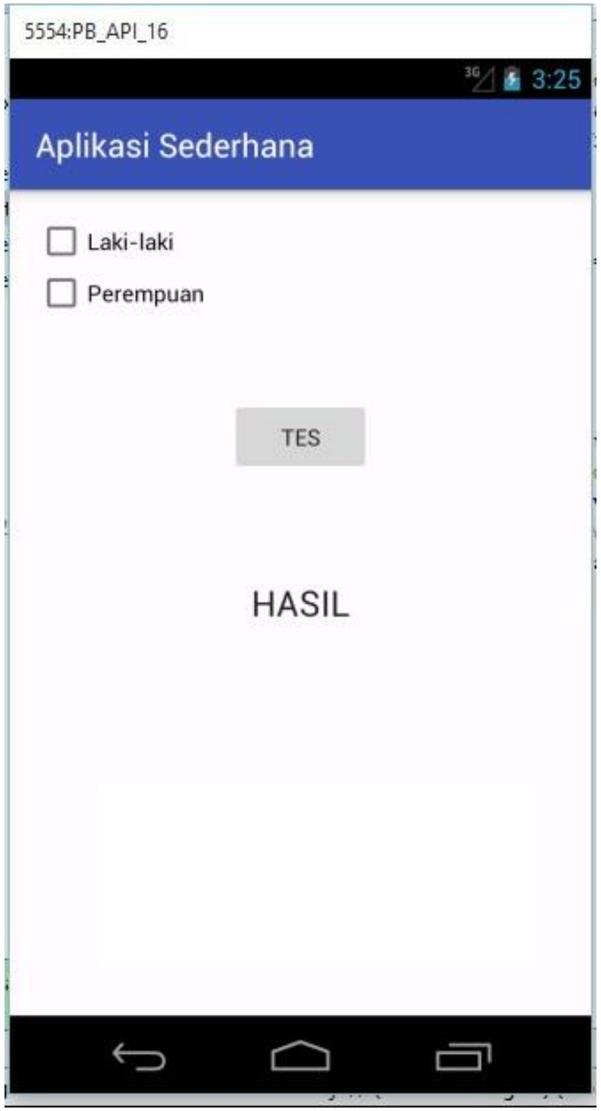
Killed – The Activity has been terminated by the runtime system in order to free up memory and is no longer present on the Activity Stack.

Android Activity Lifecycle Methods

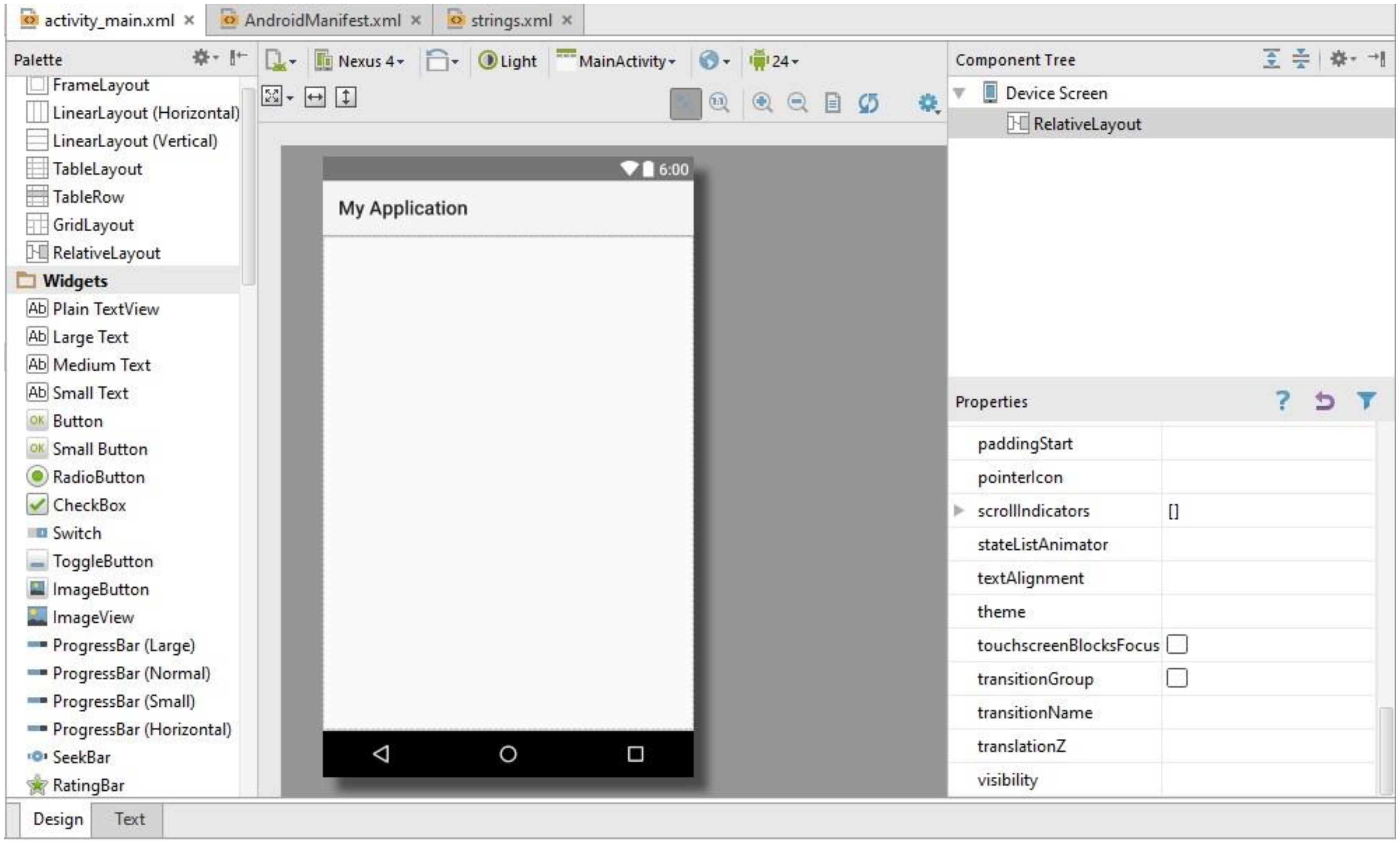


onCreate(Bundle savedInstanceState) – The method that is called when the activity is first created and the ideal location for most initialization tasks to be performed.
onRestart() – Called when the activity is about to restart after having previously been stopped by the runtime system.
onStart() – Always called immediately after the call to the *onCreate()* or *onRestart()* methods, this method indicates to the activity that it is about to become visible to the user. This call will be followed by a call to *onResume()* if the activity moves to the top of the activity stack, or *onStop()* in the event that it is pushed down the stack by another activity.
onResume() – Indicates that the activity is now at the top of the activity stack and is the activity with which the user is currently interacting
onPause() – Indicates that a previous activity is about to become the foreground activity.
onStop() – The activity is now no longer visible to the user.
onDestroy() – The activity is about to be destroyed

Example of Simple app

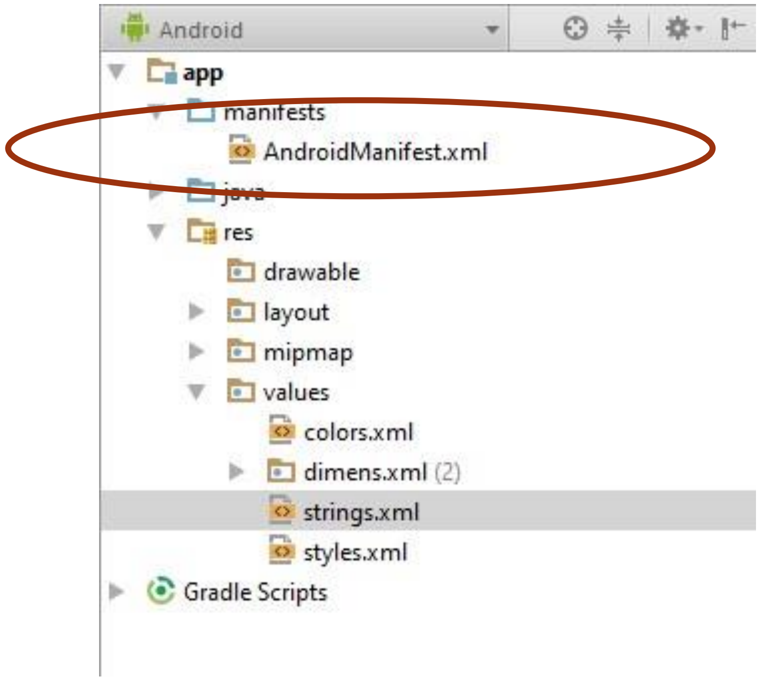


How about changing the app title?



Property	Value
paddingStart	
pointerIcon	
scrollIndicators	[]
stateListAnimator	
textAlignment	
theme	
touchscreenBlocksFocus	☐
transitionGroup	☐
transitionName	
translationZ	
visibility	

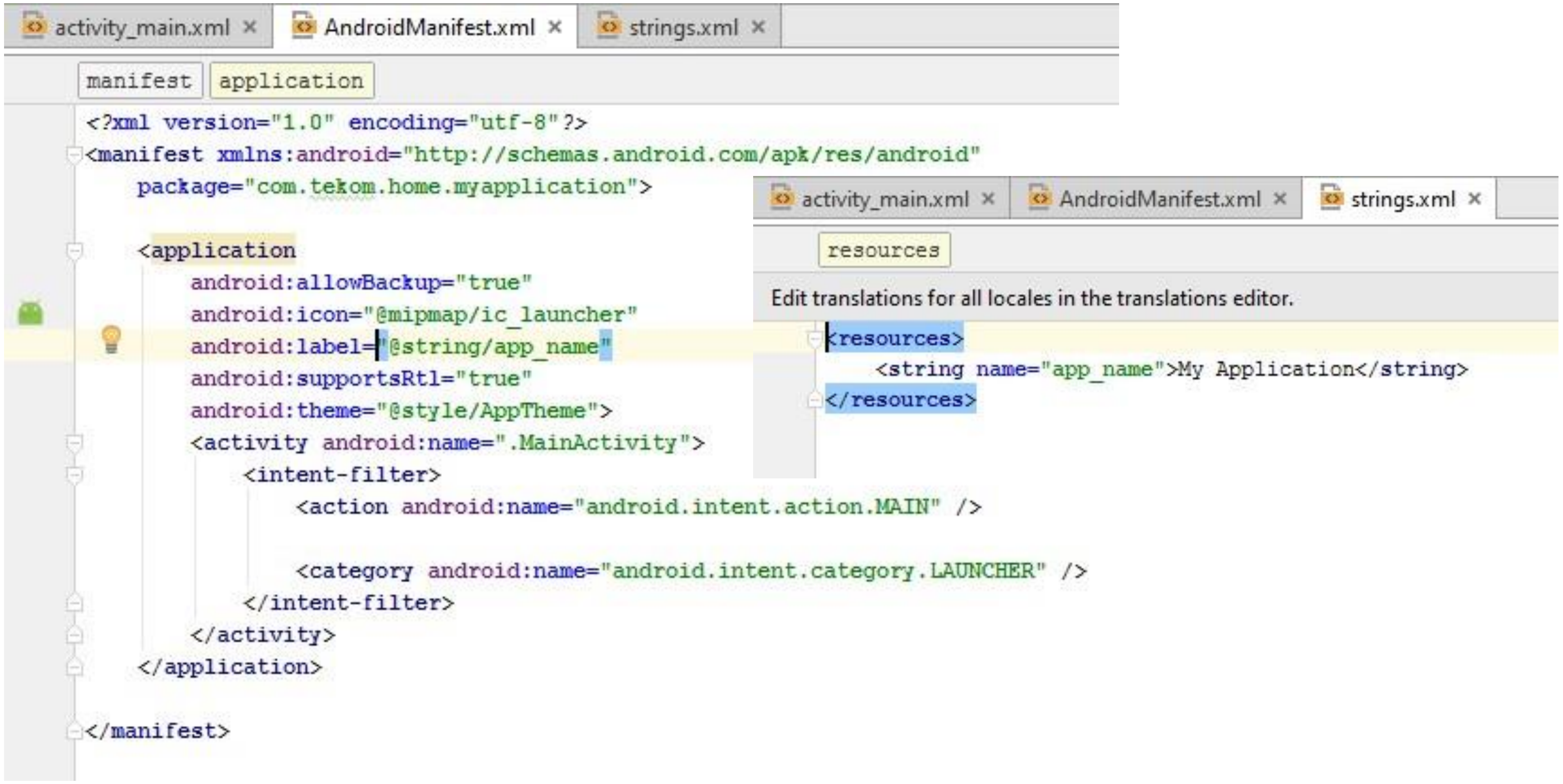
Navigate to your androidManifest.xml
Do you remember its functionality?



You'll find android:label="@string/app_name

You can change this app_name in hardcoded way, but let's try the other way around

Navigate to your strings.xml and rename your app_name there

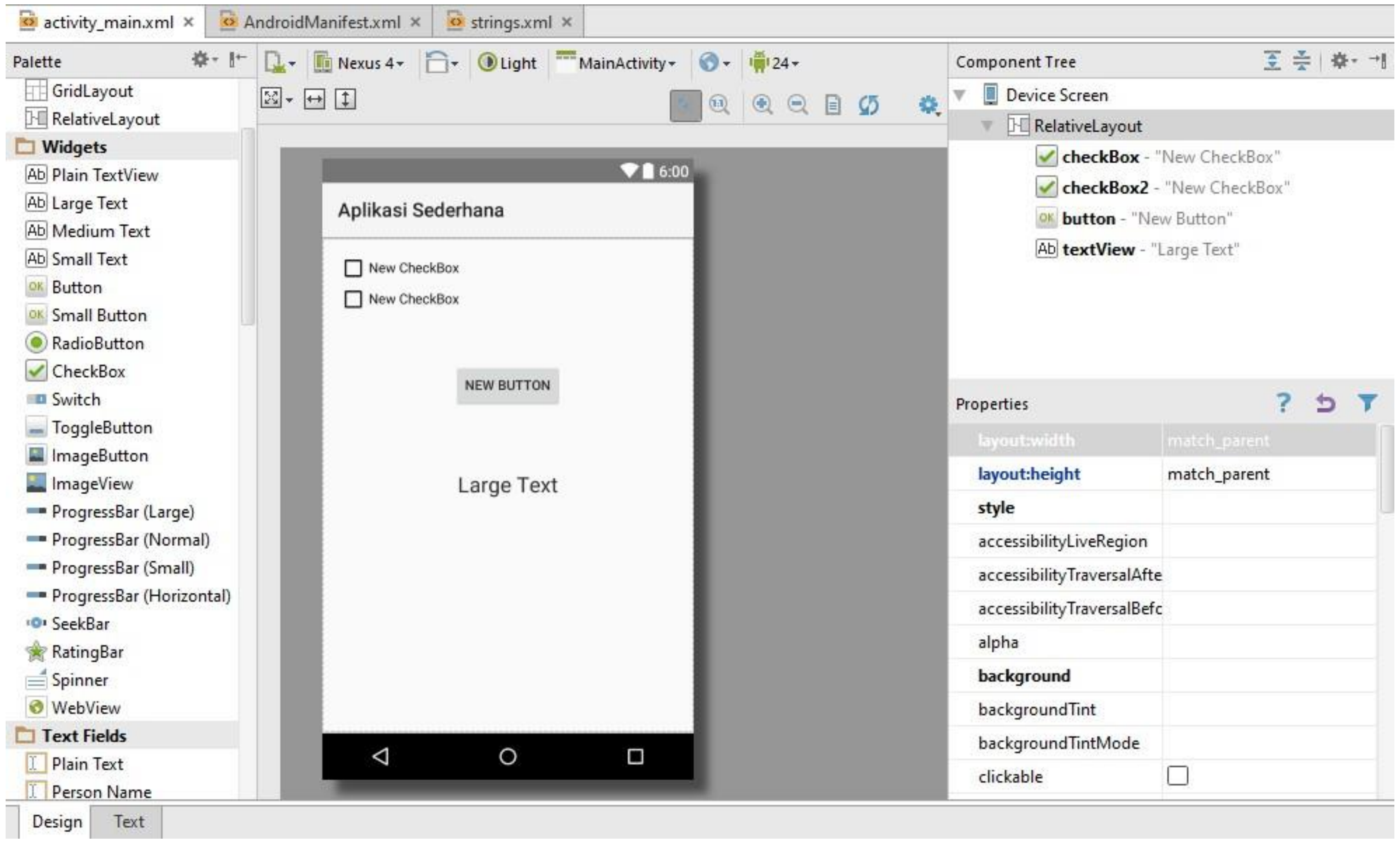


```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tekom.home.myapplication">

    <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>
```

Edit translations for all locales in the translations editor.

```
<resources>
    <string name="app_name">My Application</string>
</resources>
```

Let’s add more widgets



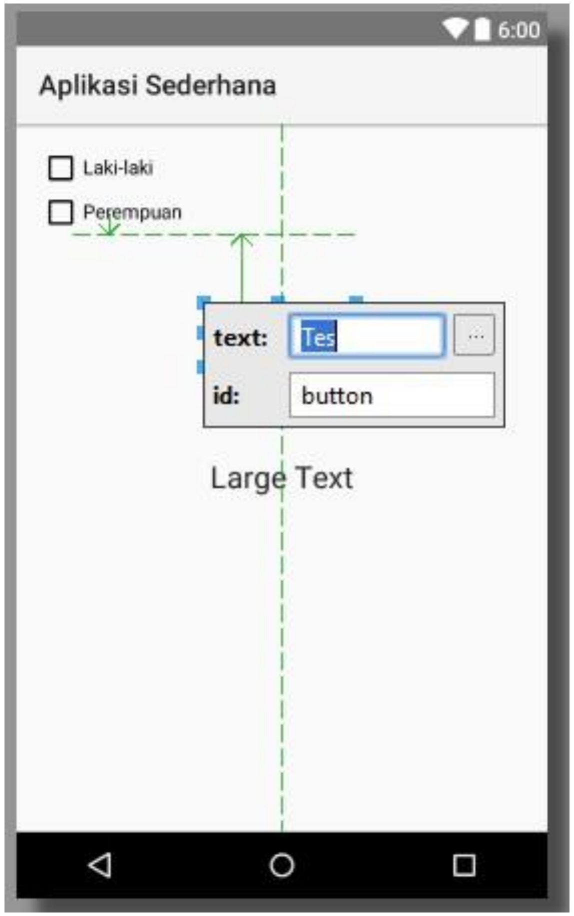
Component Tree

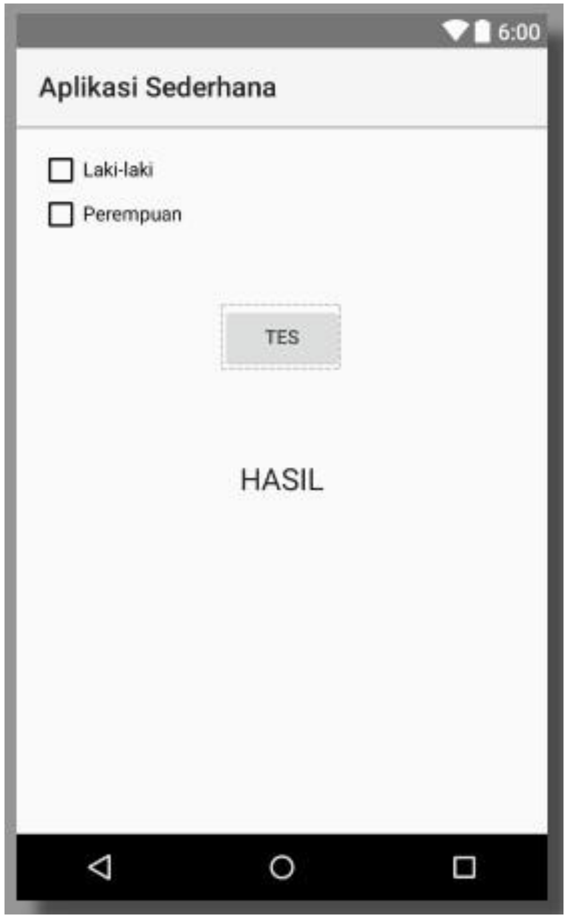
- Device Screen
  - RelativeLayout
    - **checkBox** - "New CheckBox"
    - **checkBox2** - "New CheckBox"
    - **button** - "New Button"
    - **textView** - "Large Text"

Properties

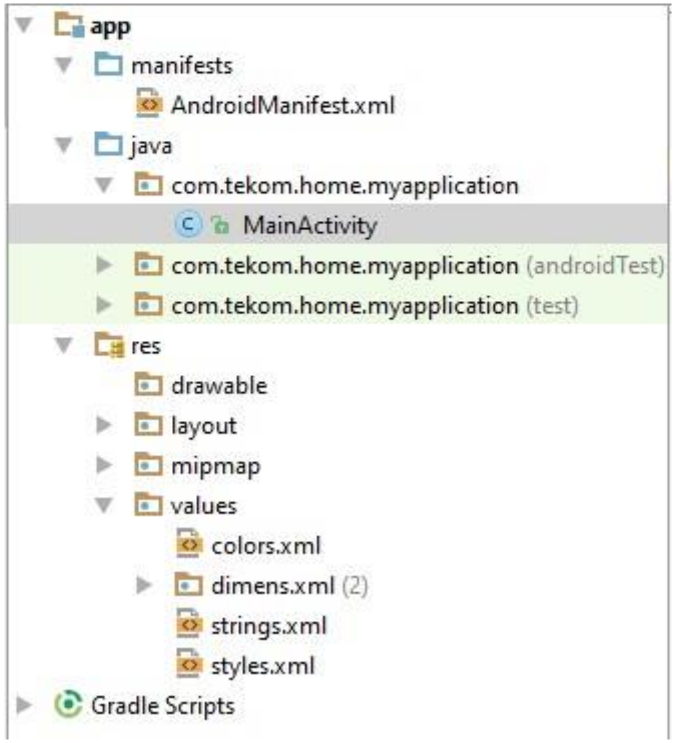
layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveRegion	
accessibilityTraversalAfte	
accessibilityTraversalBefc	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	☐

Modify the each label in hardcoded way
You can also modify via strings.xml to make locale

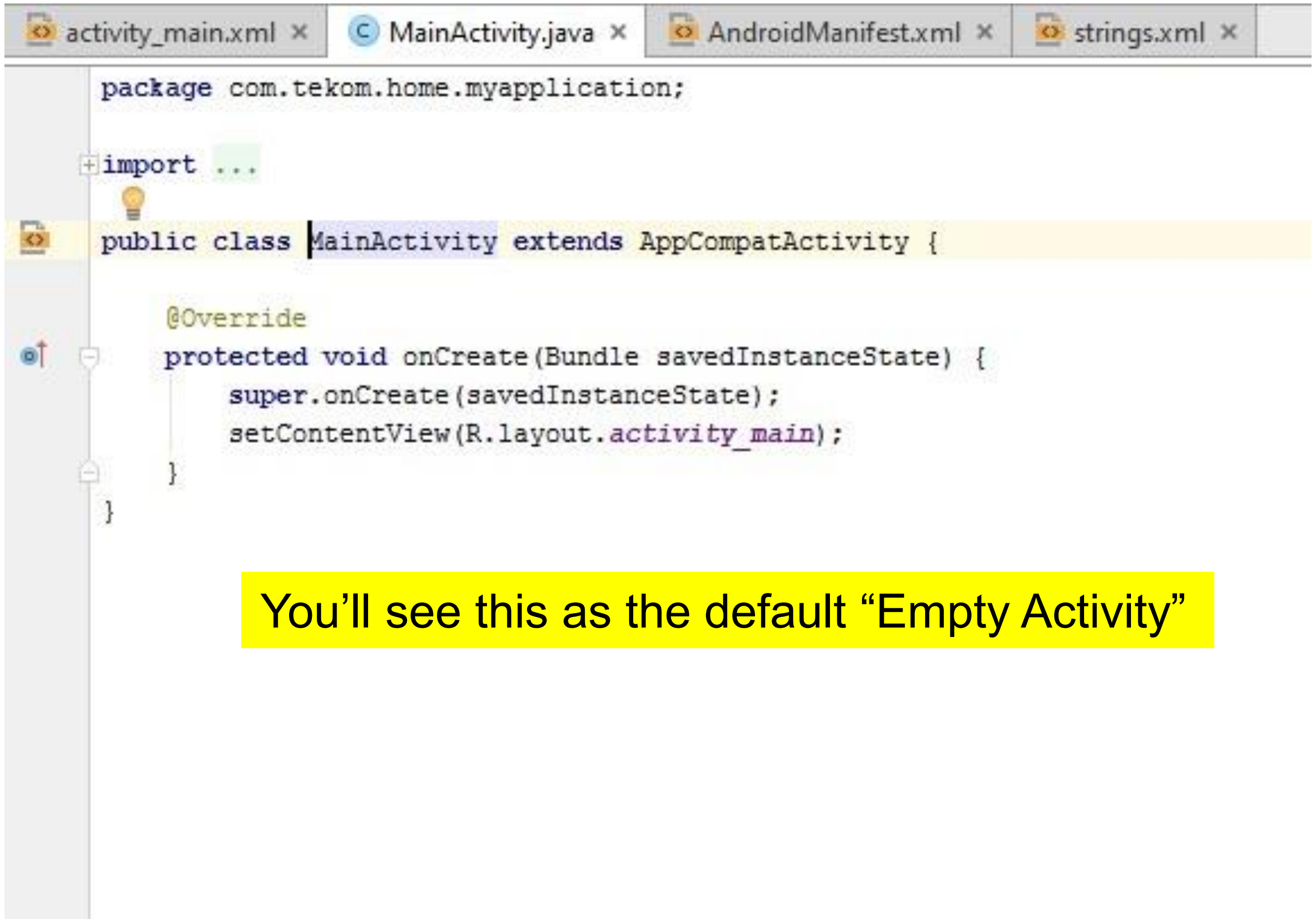




Now modify the java file to give the functionality


```
▼ app
  ▼ manifests
      AndroidManifest.xml
  ▼ java
    ▼ com.tekom.home.myapplication
        MainActivity
    ▶ com.tekom.home.myapplication (androidTest)
    ▶ com.tekom.home.myapplication (test)
  ▼ res
      drawable
    ▶ layout
    ▶ mipmap
    ▼ values
        colors.xml
      ▶ dimens.xml (2)
        strings.xml
        styles.xml
▶ Gradle Scripts
```

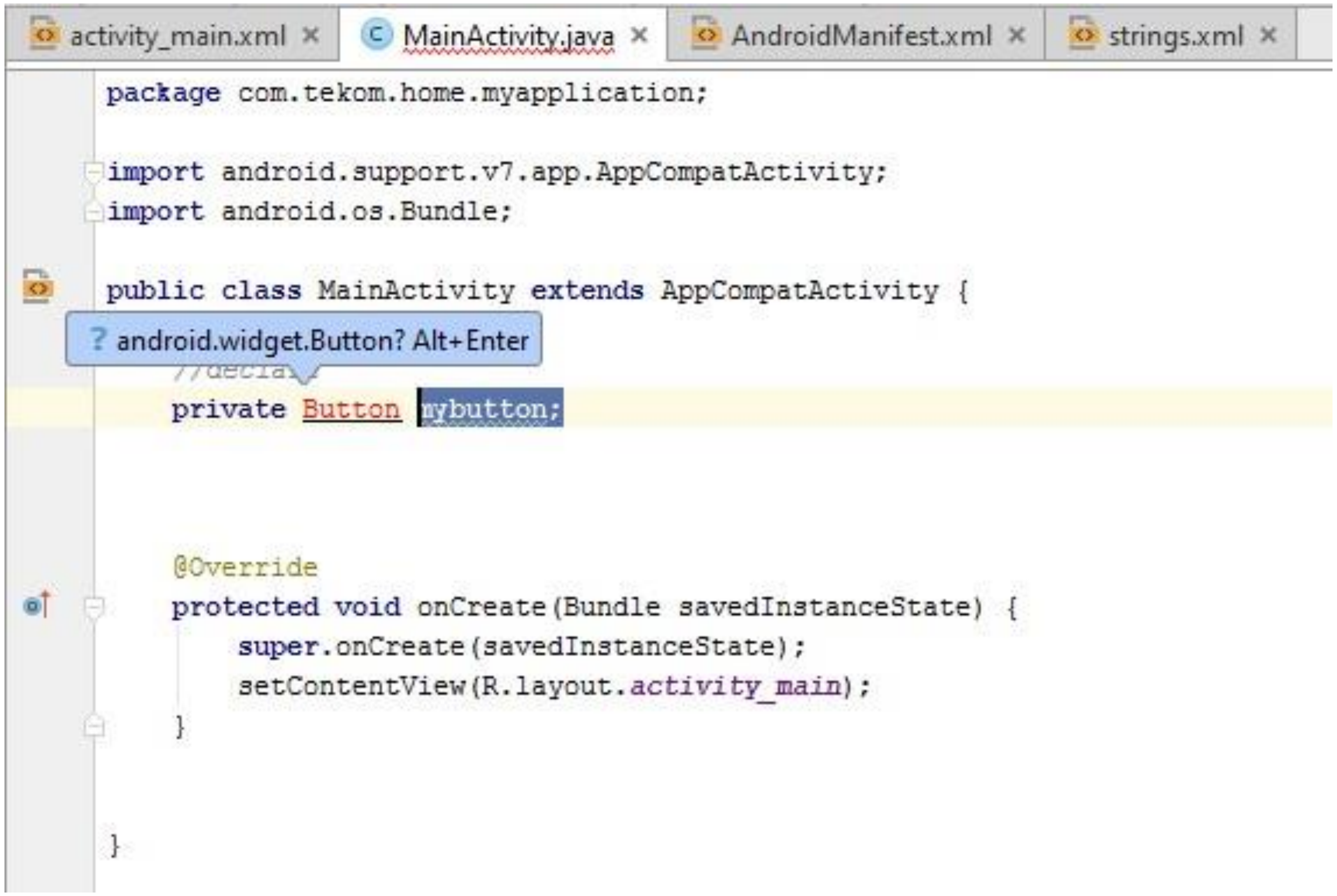
Mainactivity.java



```
package com.tekom.home.myapplication;

import ...

public class MainActivity extends AppCompatActivity {

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

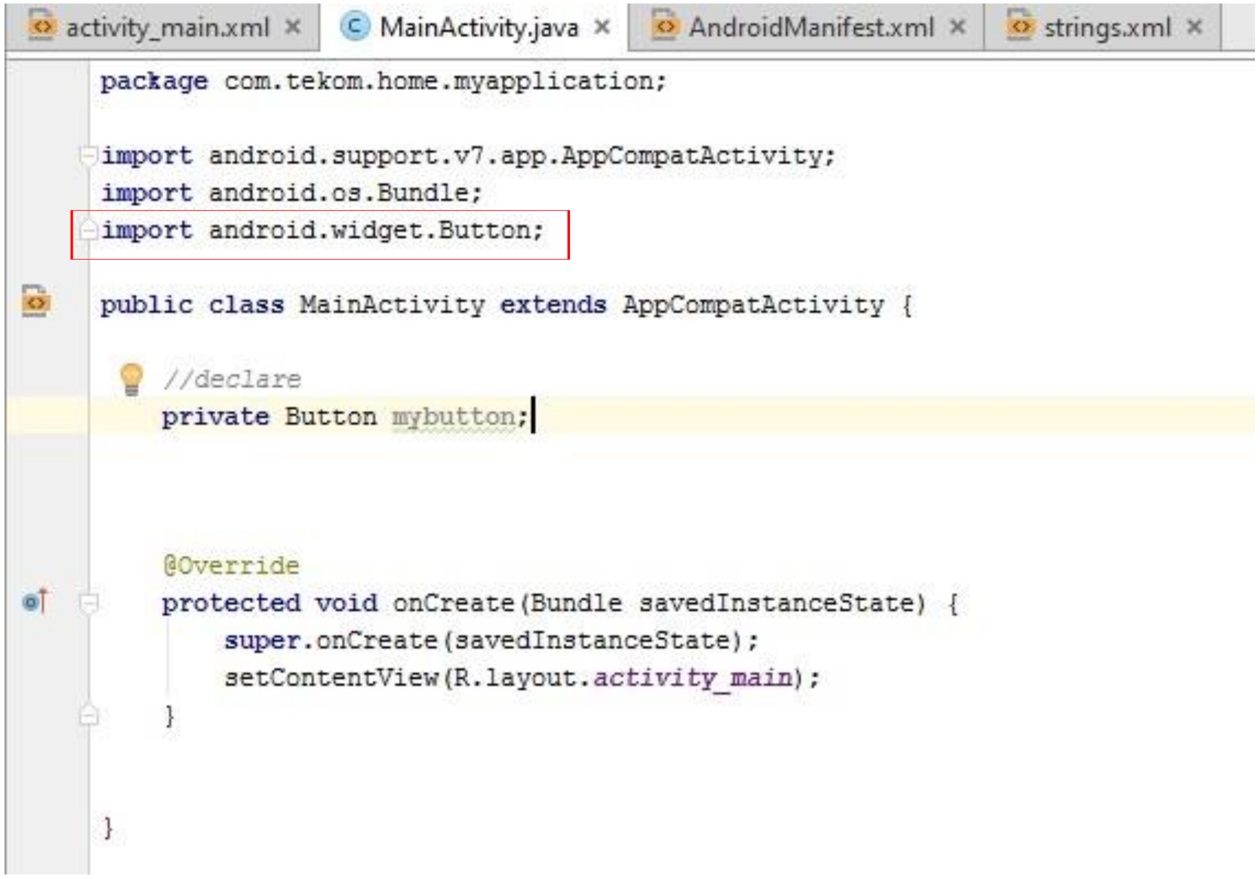
You'll see this as the default "Empty Activity"

Define your widgets, and your variable if any



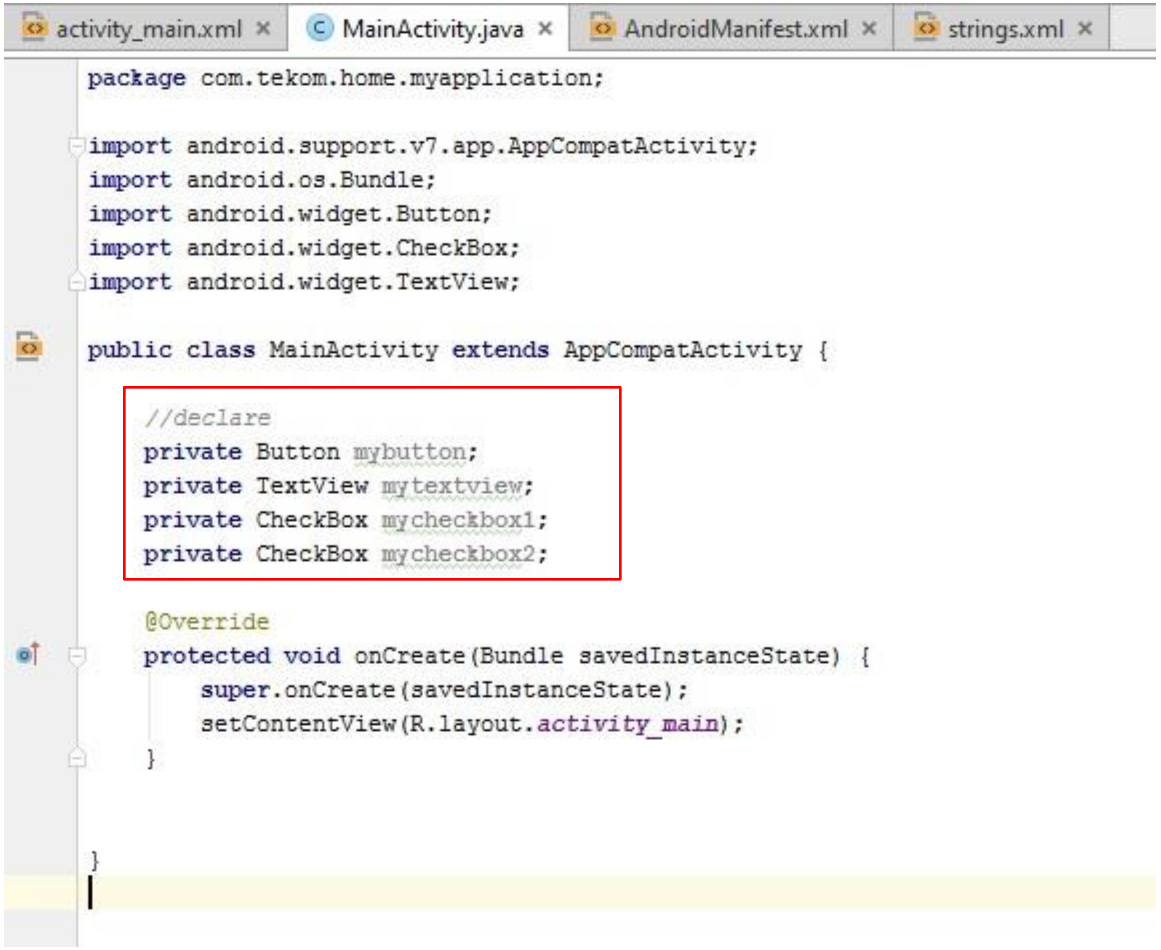
```
package com.tekom.home.myapplication;

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

public class MainActivity extends AppCompatActivity {

    //declar
    private Button mybutton;



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


}
```

Push ALT+ENTER when prompted, to correct and add to “import”



```
package com.tekom.home.myapplication;

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

public class MainActivity extends AppCompatActivity {

    //declare
    private Button mybutton;



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


}
```

Do to all widgets and variable, if any



activity_main.xml × | MainActivity.java × | AndroidManifest.xml × | strings.xml ×

```
package com.tekom.home.myapplication;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    //declare
    private Button mybutton;
    private TextView mytextview;
    private CheckBox mycheckbox1;
    private CheckBox mycheckbox2;

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


}
```

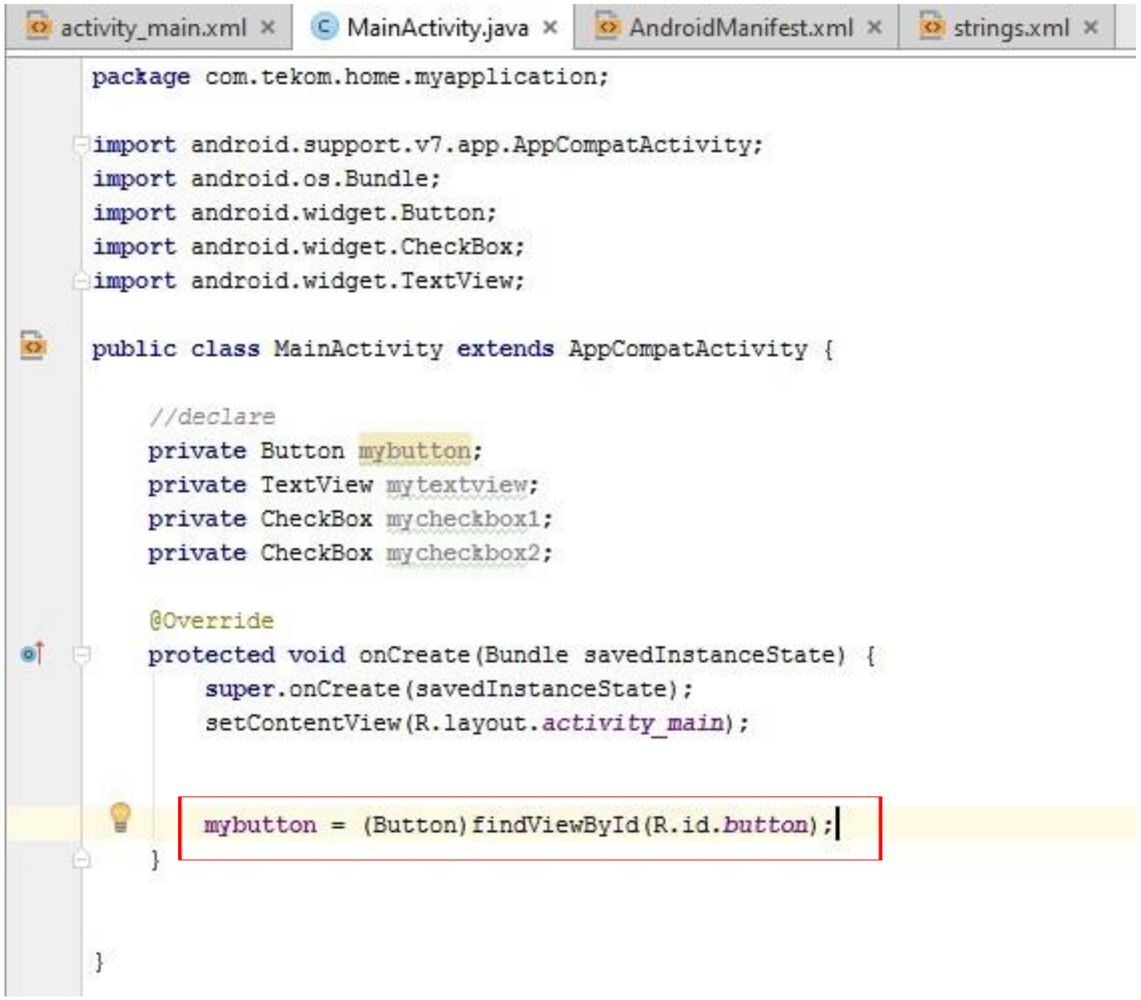
Adding Widget Objects to Activity

Before widgets and views are manipulatablеs through Activity, widgets and views have to be loaded into Activity using:

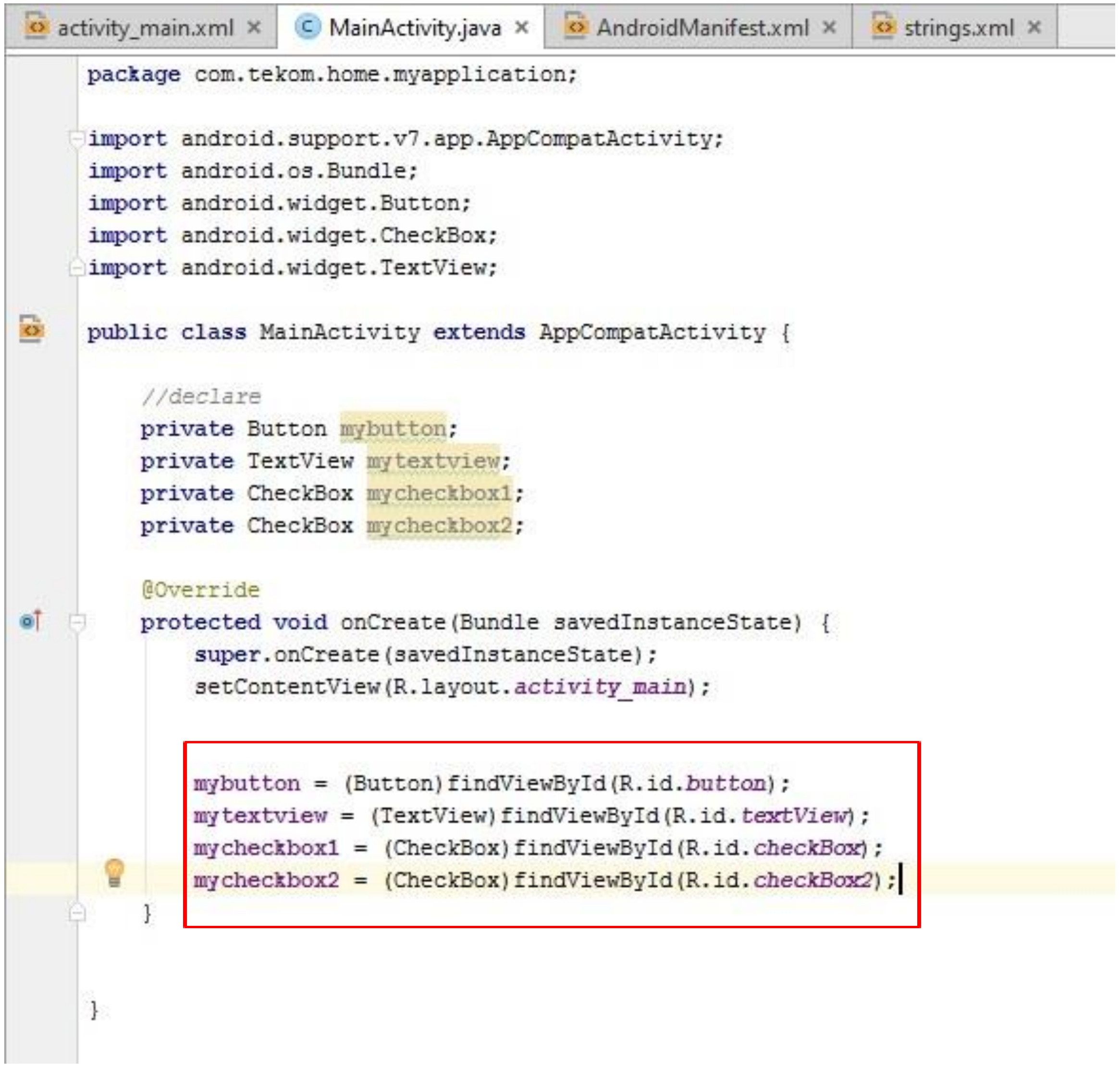
- `View v = findViewById([widget-resource-id])`

[widget-resource-id] is defined in UI XML file, called <u>after</u> calling `setContentView()` method:

- E.g. in XML:
  ```
  <TextView android:id="@+id/nameTextView" />
  ```
- In Activity:
  ```
  TextView nameView = (TextView)
      findViewById(R.id.nameTextView);
  ```



```java
package com.tekom.home.myapplication;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    //declare
    private Button mybutton;
    private TextView mytextview;
    private CheckBox mycheckbox1;
    private CheckBox mycheckbox2;

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


        mybutton = (Button)findViewById(R.id.button);
    }


}
```



```
package com.tekom.home.myapplication;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    //declare
    private Button mybutton;
    private TextView mytextview;
    private CheckBox mycheckbox1;
    private CheckBox mycheckbox2;

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

        mybutton = (Button)findViewById(R.id.button);
        mytextview = (TextView)findViewById(R.id.textView);
        mycheckbox1 = (CheckBox)findViewById(R.id.checkBox);
        mycheckbox2 = (CheckBox)findViewById(R.id.checkBox2);
    }

}
```

Event Handling

Decide what Widgets who’s events to process

Define an event listener and register it with the View.

- View.OnClickListener (for handling "clicks" on a View), View.OnTouchListener (for handling touch screen events in a View), and View.OnKeyListener (for handling device key presses within a View)

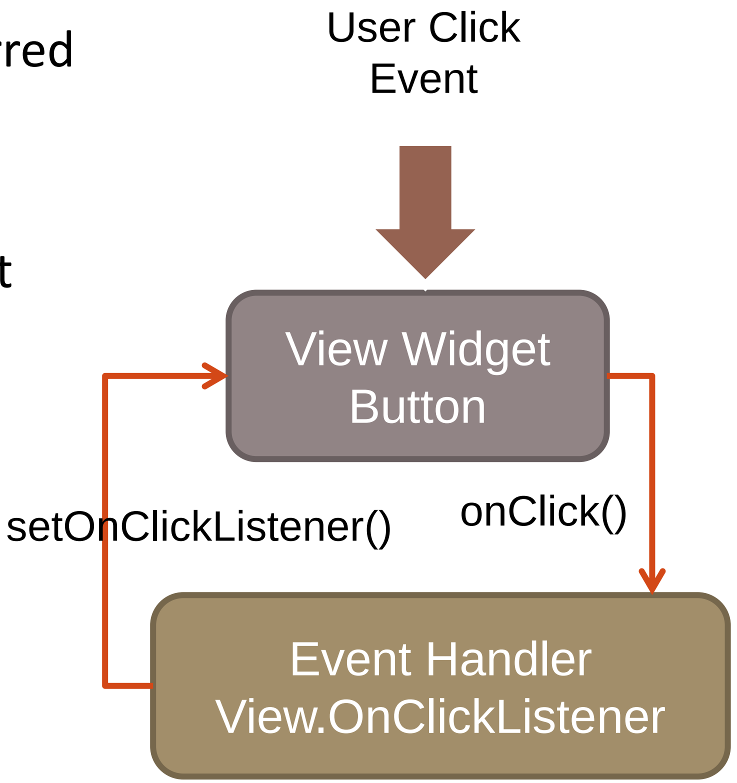
http://developer.android.com/guide/topics/ui/ui-events.html details more

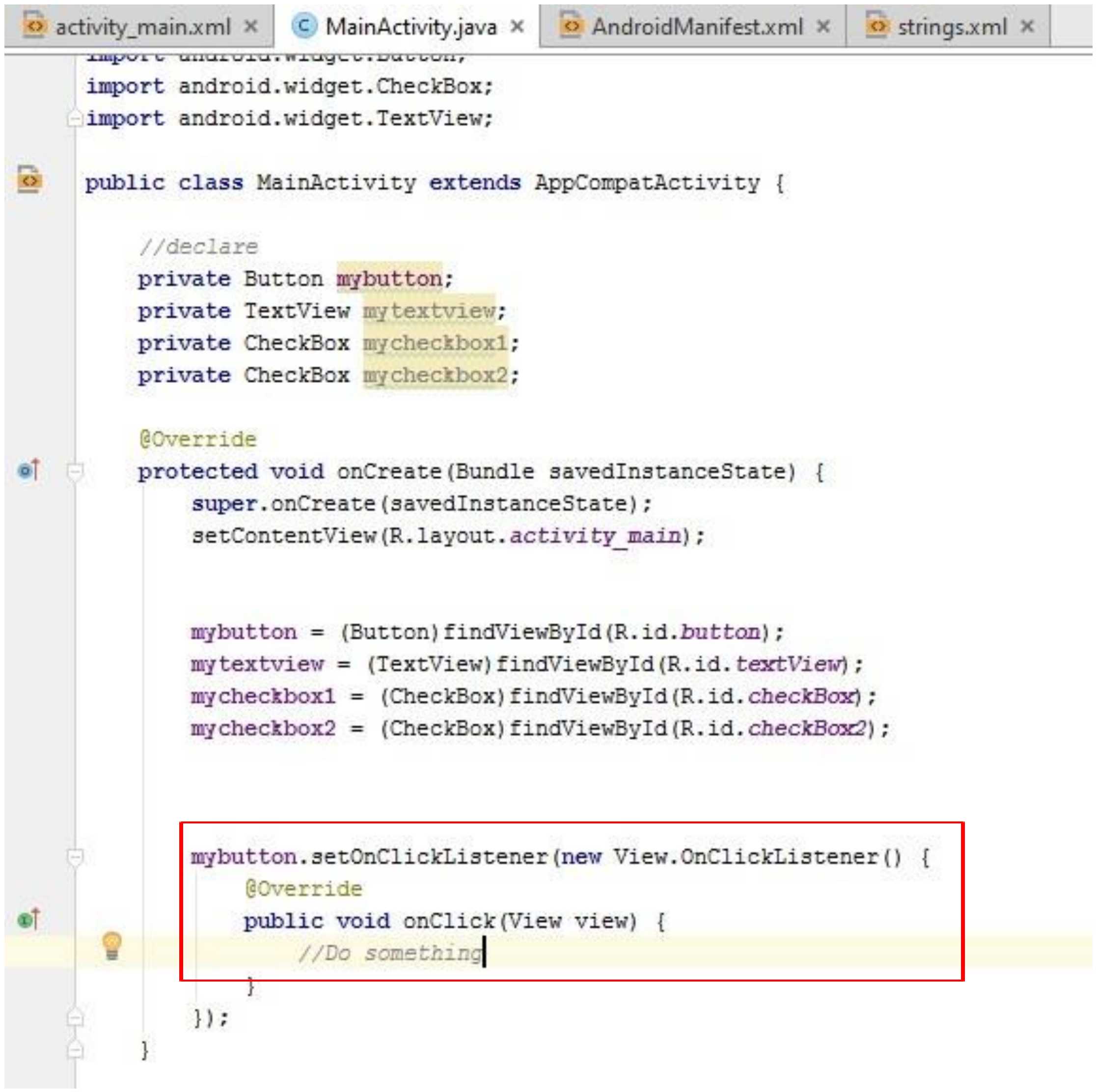
Event Handling

Event Handler is a listener object which handle an event and do some action based on what it listen to

A listener need to be set to a view or widget so that it would be able to listen to an event occurred to the view or widget it is assigned to

A listener has an action method that being executed when an event occurred to the widget

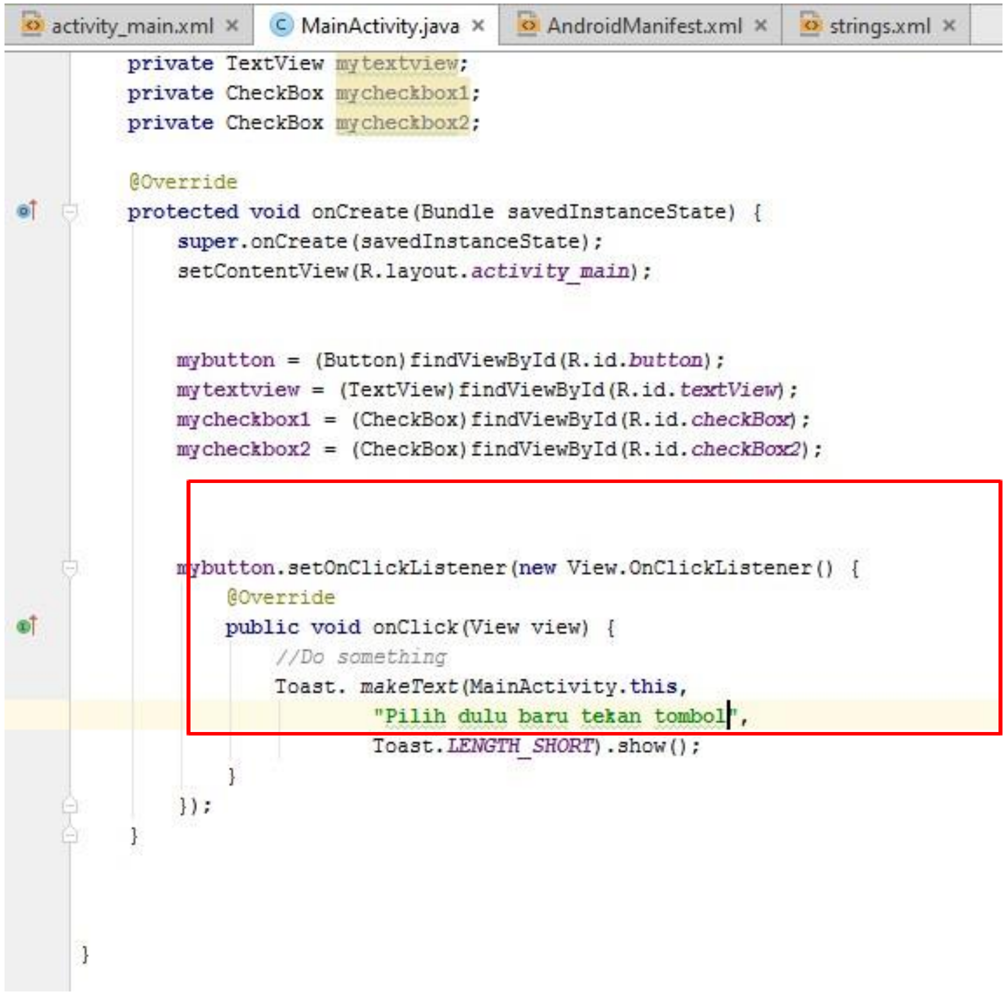




activity_main.xml × | MainActivity.java × | AndroidManifest.xml × | strings.xml ×

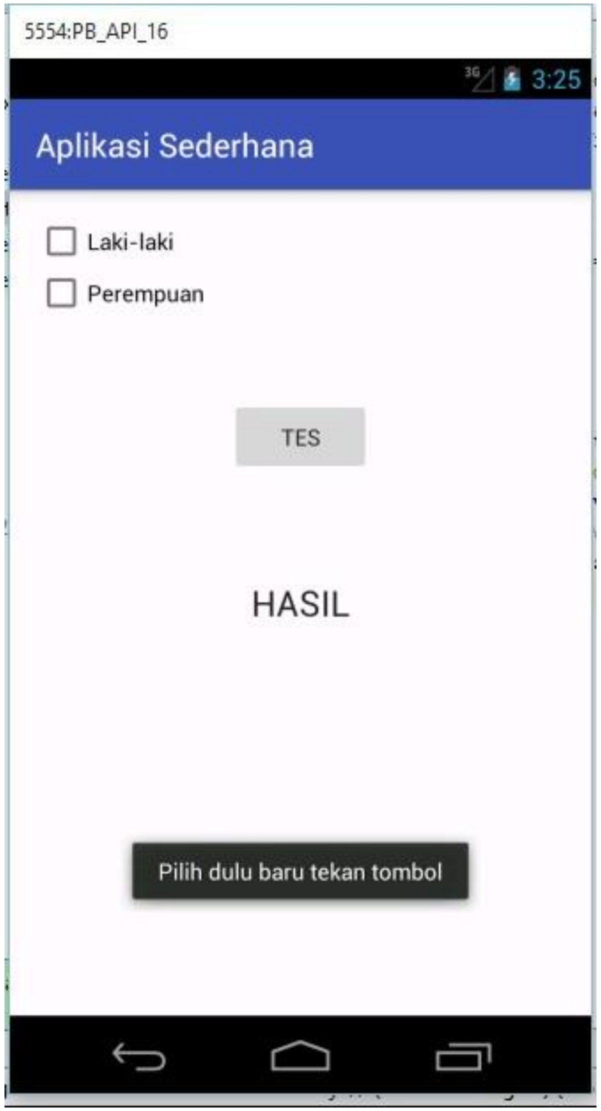
```
import android.widget.CheckBox;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    //declare
    private Button mybutton;
    private TextView mytextview;
    private CheckBox mycheckbox1;
    private CheckBox mycheckbox2;

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


        mybutton = (Button)findViewById(R.id.button);
        mytextview = (TextView)findViewById(R.id.textView);
        mycheckbox1 = (CheckBox)findViewById(R.id.checkBox);
        mycheckbox2 = (CheckBox)findViewById(R.id.checkBox2);


        mybutton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Do something
            }
        });
    }
```

Try to display **Toast Message** when we click the button


activity_main.xml × | MainActivity.java × | AndroidManifest.xml × | strings.xml ×

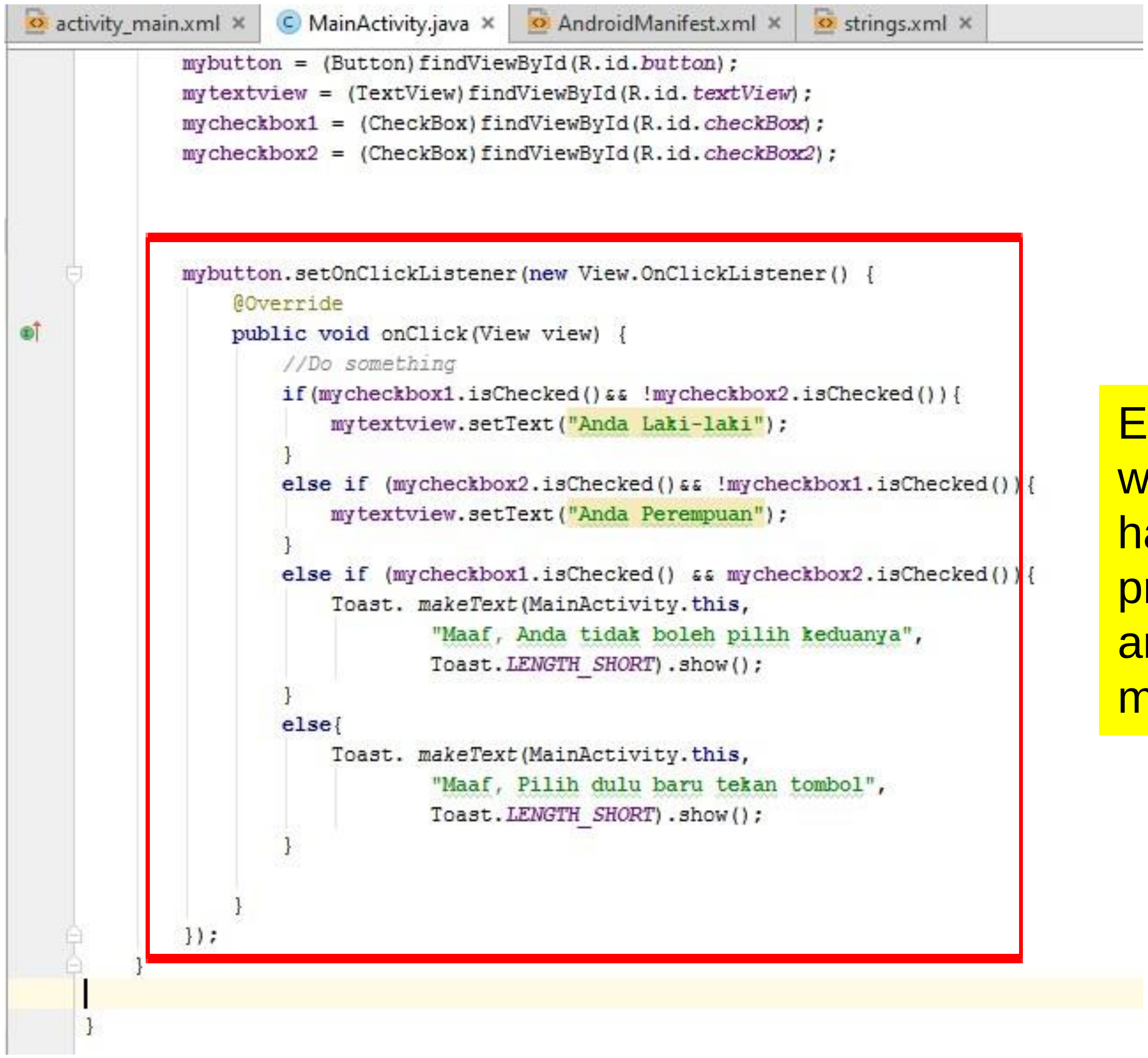
```
    private TextView mytextview;
    private CheckBox mycheckbox1;
    private CheckBox mycheckbox2;

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


        mybutton = (Button)findViewById(R.id.button);
        mytextview = (TextView)findViewById(R.id.textView);
        mycheckbox1 = (CheckBox)findViewById(R.id.checkBox);
        mycheckbox2 = (CheckBox)findViewById(R.id.checkBox2);



        mybutton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Do something
                Toast. makeText(MainActivity.this,
                        "Pilih dulu baru tekan tombol",
                        Toast.LENGTH_SHORT).show();
            }
        });
    }



}
```

Result:
Toast was displayed when we click the button


5554:PB_API_16

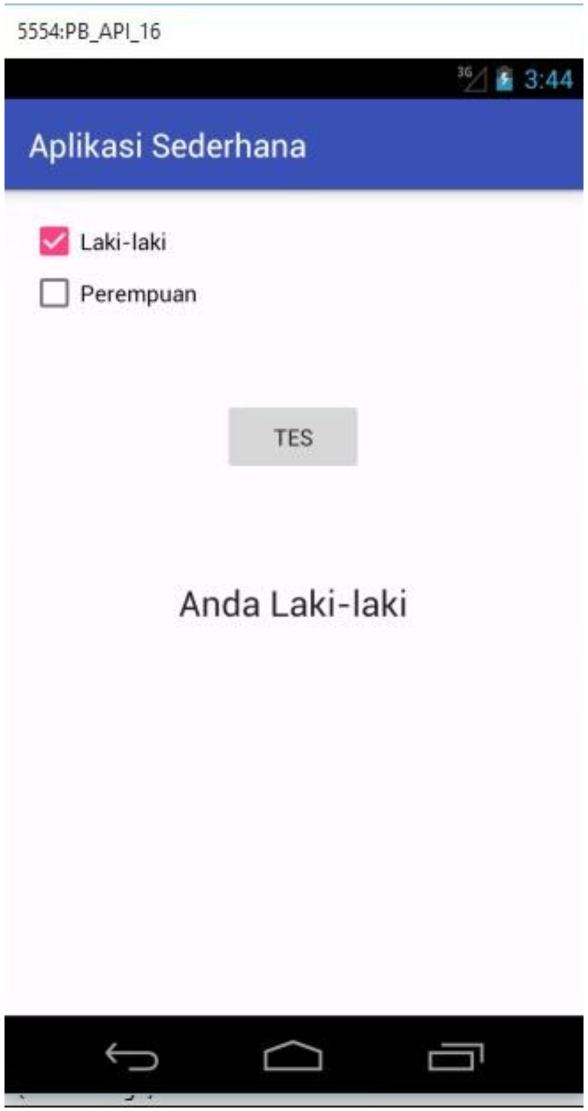
Pilih dulu baru tekan tombol

Complete the programming ..


activity_main.xml × MainActivity.java × AndroidManifest.xml × strings.xml ×

```
        mybutton = (Button)findViewById(R.id.button);
        mytextview = (TextView)findViewById(R.id.textView);
        mycheckbox1 = (CheckBox)findViewById(R.id.checkBox);
        mycheckbox2 = (CheckBox)findViewById(R.id.checkBox2);


        mybutton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Do something
                if(mycheckbox1.isChecked()&& !mycheckbox2.isChecked()){
                    mytextview.setText("Anda Laki-laki");
                }
                else if (mycheckbox2.isChecked()&& !mycheckbox1.isChecked()){
                    mytextview.setText("Anda Perempuan");
                }
                else if (mycheckbox1.isChecked() && mycheckbox2.isChecked()){
                    Toast. makeText(MainActivity.this,
                            "Maaf, Anda tidak boleh pilih keduanya",
                            Toast.LENGTH_SHORT).show();
                }
                else{
                    Toast. makeText(MainActivity.this,
                            "Maaf, Pilih dulu baru tekan tombol",
                            Toast.LENGTH_SHORT).show();
                }

            }
        });
    }

}
```

Every widget/view has its own properties and methods!

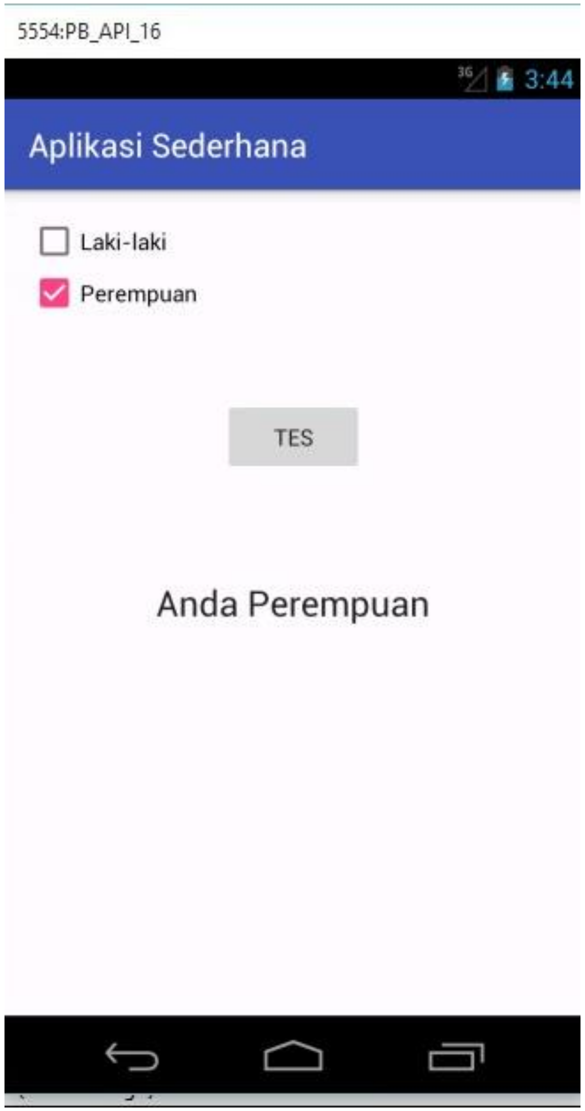

5554:PB_API_16

3:44

Aplikasi Sederhana

- [✓] Laki-laki
- [ ] Perempuan

Anda Laki-laki

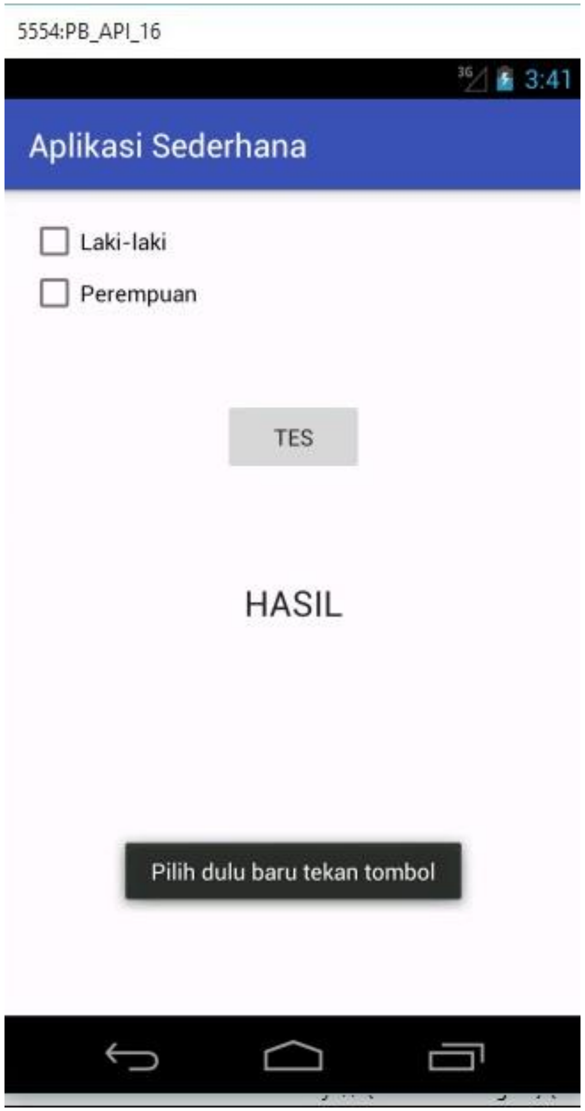

5554:PB_API_16

3:44

Aplikasi Sederhana

- [ ] Laki-laki
- [✓] Perempuan

Anda Perempuan


5554:PB_API_16

3:41

Aplikasi Sederhana

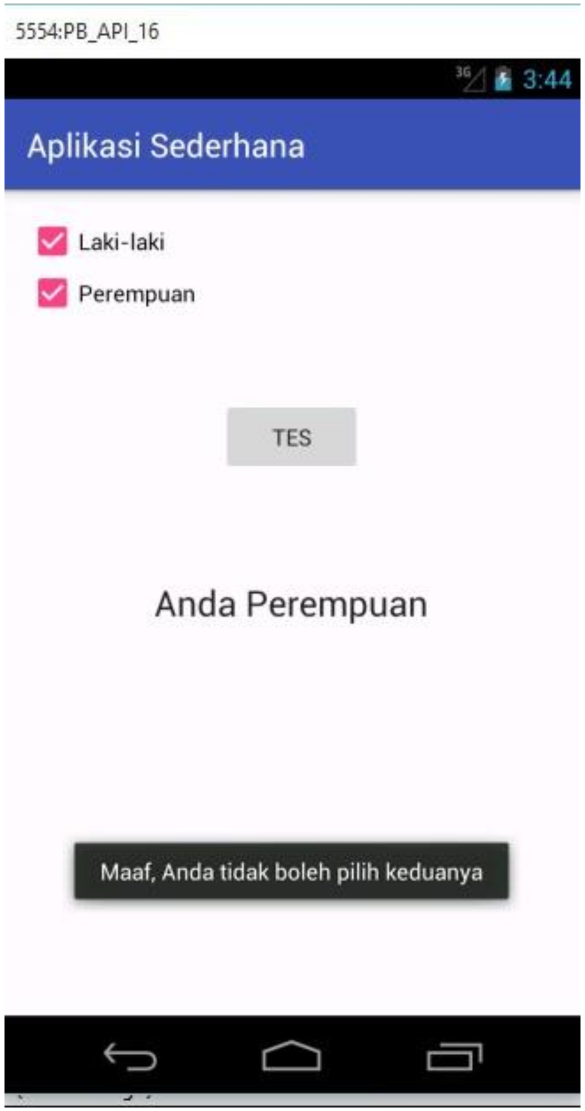
☐ Laki-laki

☐ Perempuan

TES

HASIL

Pilih dulu baru tekan tombol


5554:PB_API_16

3:44

Aplikasi Sederhana

☑ Laki-laki

☑ Perempuan

TES

Anda Perempuan

Maaf, Anda tidak boleh pilih keduanya

TERIMA KASIH