

COMP3033J Mobile Application Development

Android Internals

Dr. Abraham Campbell

University College Dublin

Abey.campbell@ucd.ie

Outline

- Typical Handset Capabilities
- Android vs Java
- Android architecture
- Building blocks of Android application
- Android project file structure
- App compilation and deployment

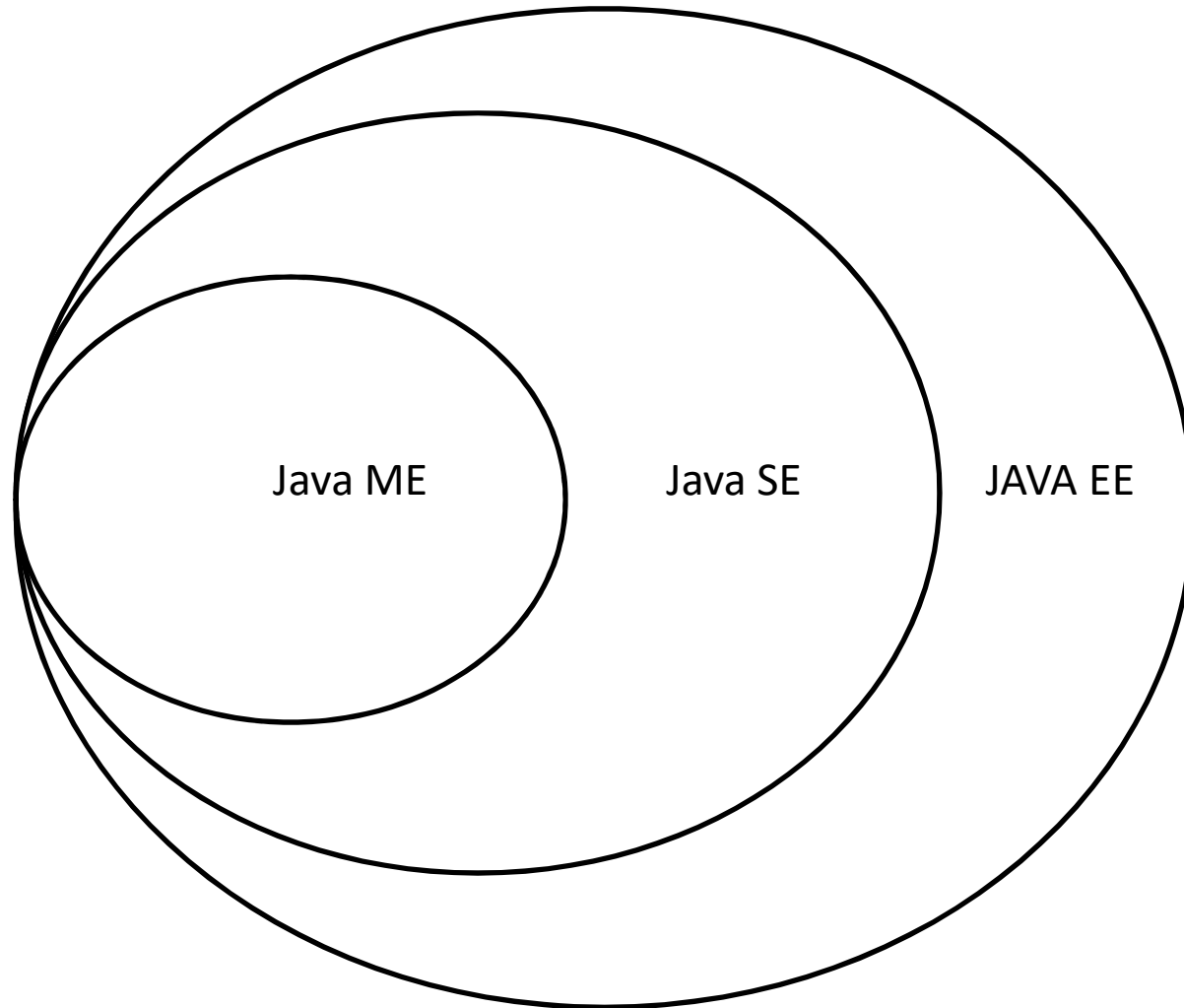
Typical Android-powered Handset

- **Phone Services: GSM (CDMA), SMS**
- **Networking: Wi-Fi, WiMAX, 2G (GPRS, EDGE), 3G (UMTS,HSPA), 4G(LTE, HSPA+), Bluetooth, NFC**
- **Location services: GPS, AGPS, GLONASS, etc...**
- **Multimedia hardware: photo, video camera(s)/microphone**
- **Positioning: accelerometer, compass, gyroscope, barometer, magnetometer**
- **Web Browser: WebKit-based (now Chromium)**
- **Graphics: hardware-accelerated, 2D and 3D**
- **Storage / Encrypted storage**

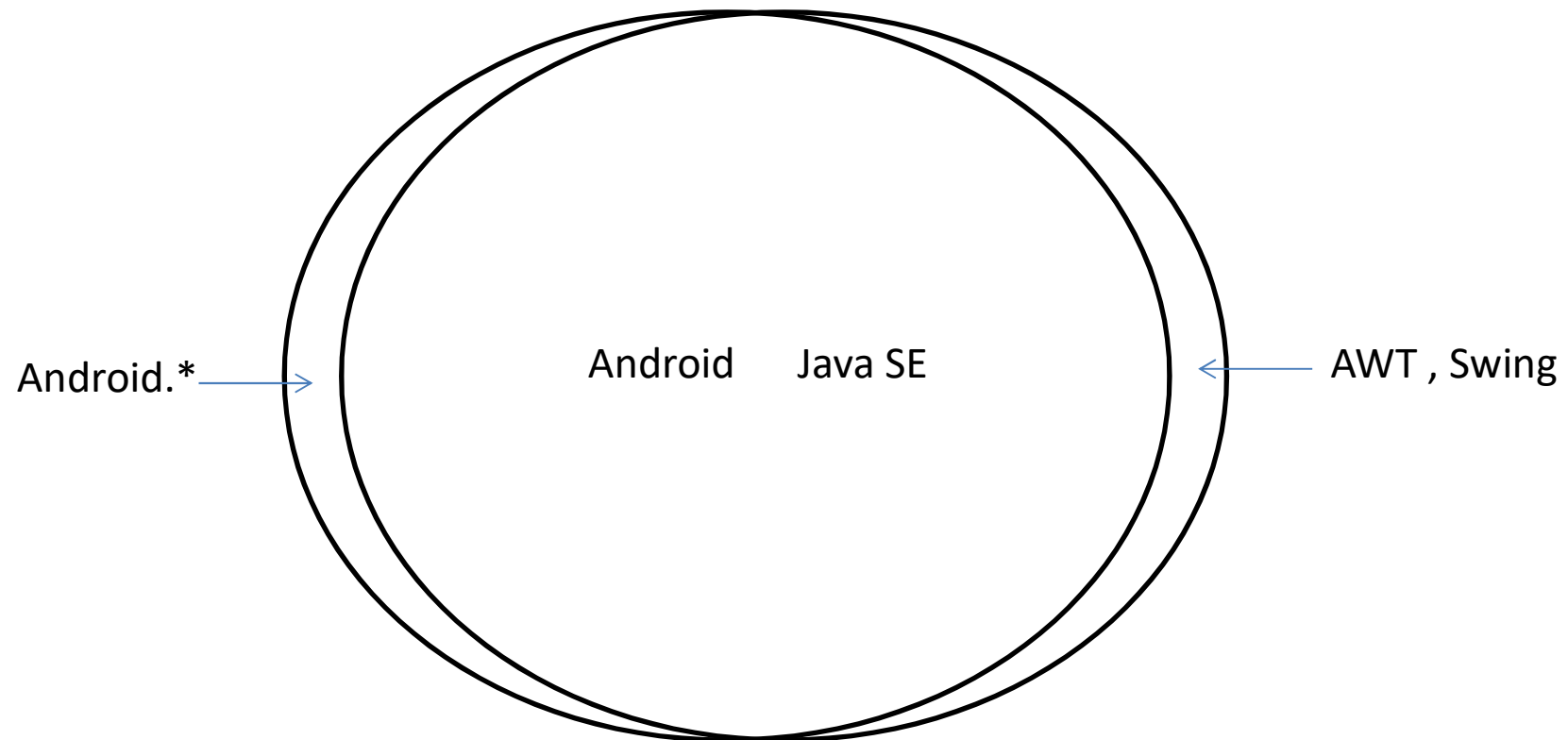
Android vs Java

- Android is a mobile operating system
- Java is a programming language for various platforms
- Android programming is done with Java compatible libraries based on (now deprecated) Apache Harmony project

Java Platform Variations



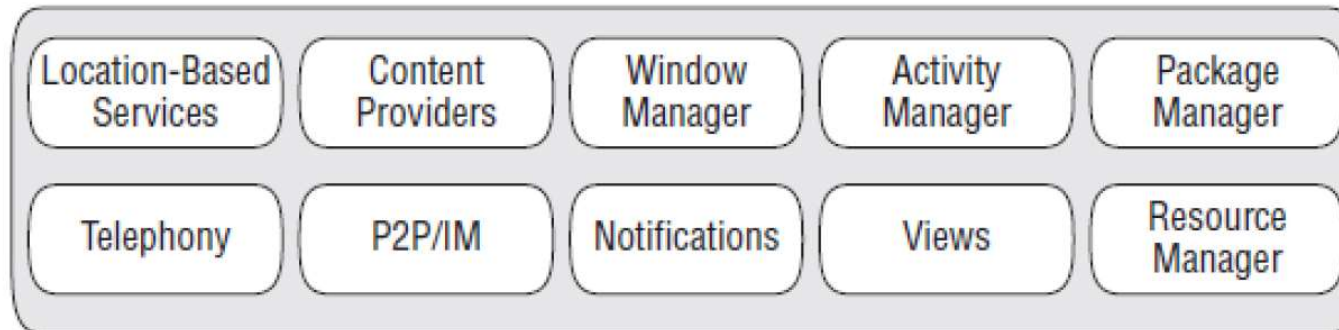
Android vs Java



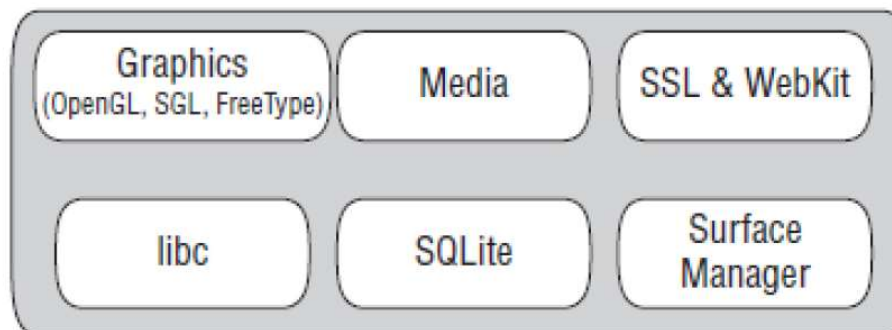
Application Layer



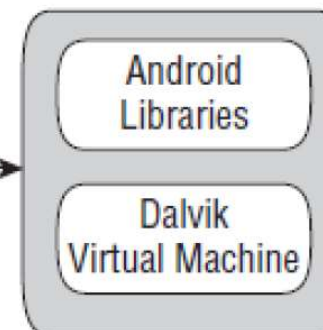
Application Framework



Libraries



Android Runtime



Linux Kernel

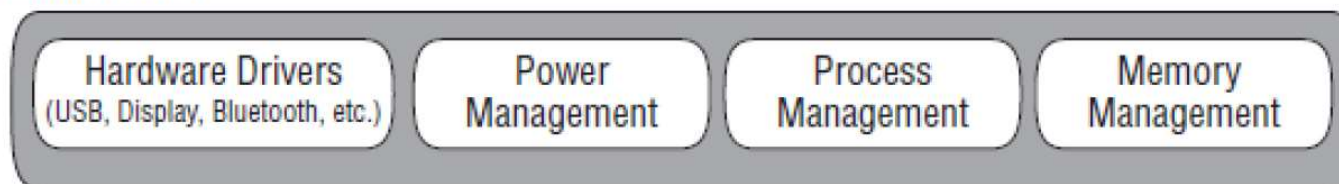


Figure 1-1

(c) Reto Meier

Process/Thread Perspective

- Security is managed by the kernel
 - Each app runs in its own process
 - Each app gets its own unique UID/GID
 - Each app runs its own Dalvik VM
-
- The processes are isolated and crash of one app does not bring down the whole system!

Android.*

- android.util – containers, formatters, parsers
- android.os – message passing, IPC, debugging
- android.graphics – canvas, colour, primitives, ...
- android.text – displaying & parsing text
- android.database – handling cursors for db
- android.content – data access and publishing
- android.view – core user interface
- android.widget – lists, buttons, layouts...

Android.* Cont'd

- `com.google.android.maps` – map controls
- `android.app` – activity and service API
- `android.provider` – standard content provider
- `android.telephony` – telephony API
- `android.webkit` – web-based content work
- Also: OpenGL, FreeType, SGL, libc, SQLite, SSL

Android.* Cont'd

- android.location
- android.media
- android.opengl
- android.hardware
- android.bluetooth
- android.net.wifi
- Android.nfc
- android.print

Android.* Cont'd

- android.speech
- Android.animation
- These are just the main ones. Please look at the SDK docs for more info
- I will show you how to use the doc's in class.
- I will have them on usb stick and we will pass around the copies around class.

Supported Media Formats

- Audio: 3gp, mp4, aac, ts, flac, mid, ogg, mkv, wav
- Images: jpeg, gif, png, bmp, *webp (v4)*
- Video: 3gp, mp4, ts, *webm, mkv*

Basic Building Blocks

- Activities
- Content Providers
- Services
- Intents

Activities

- The building block of the user interface is the activity. You can think of an activity as being the Android analogue for the window or dialog box in a desktop application.

Content Providers

- Content providers provide a level of abstraction for any data stored on the device that is accessible by multiple applications.
- The Android development model encourages you to make your own data available to other applications, as well as your own. Building a content provider lets you do that, while maintaining complete control over how your data is accessed.

Services

- Services are designed to keep running, if needed, independent of any activity. You might use a service for checking for updates to an RSS feed or to play back music even if the controlling activity is no longer operating.

Intents

- Intents are system messages, running around the inside of the device, notifying applications of various events, from hardware state changes, to incoming data, to application events.
- Not only can you respond to intents, but you can create your own to launch other activities or to let you know when specific situations arise.

Basics in the Nutshell

- Activity: basic building block of an application
- Intents: communication mechanism
- Service: background process with no user interface
- Content provider: basic superclass framework for handling and storing data

Android Project Structure

- **AndroidManifest.xml**
- build.xml
- default.properties, local.properties
- assets/
- bin/
- gen/
- libs/
- **src/**
- **res/**
- tests/

AndroidManifest.xml

- AndroidManifest.xml is a foundation of any Android application
- Contains a list of application activities, services
- Describes how particular app fits into the rest of Android system (system menus, etc)
- Requirements to run (sdk version)
- Describes required and provided permissions, libraries, etc.

AndroidManifest.xml

```
<manifest  
  xmlns:android="http://schemas.android.com/apk/r  
    es/android"  
  package="ie.ucd.sampleapp">  
  <usespermission... />  
  <uses sdk... />  
  <useslibrary... />  
  <application>...</application>  
</manifest>
```

src/

- Contains source code of your app in
src/ie/ucd/ex1/SampleApp.java

...

src/ie/ucd/ex1/SampleActivity.java

res/

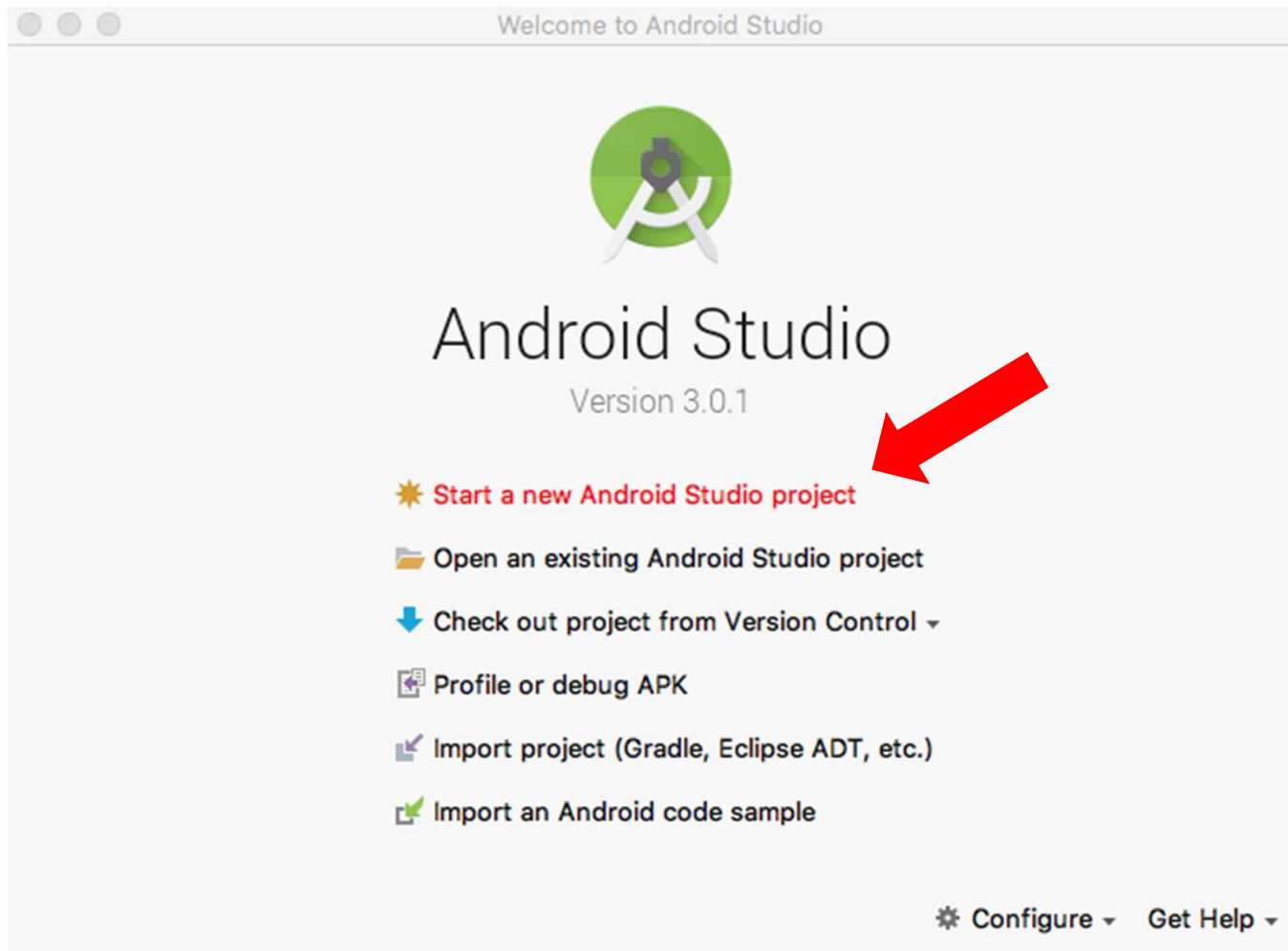
- Res directory holds static resources for your app that are packaged along with your application:
 - drawable/
 - layout/
 - menu/
 - raw/
 - values/
 - xml/

Compile Time!

- .class files – Java bytecode
- .dex – Android executable for Dalvik VM –highly optimised Java bytecode
- Yourapp-debug.apk, yourappunsigned.apk – .dex with resources, ready to debug/deploy
- New Android 5.0 (2014) creates – Android executables for the ART(Android runtime) VM
- Android 7.0 (2016) supports Daydream Google new VR platform , VULCAN api for graphics and Open-JDK for its compile and runtime
- Android 8.0 (2017) OREO supports Google ARCORE



Creating an Android project Android Studio





Creating an Android project Android Studio



Create New Project

Create Android Project

Application name
HelloAndriod

Company domain
colombo.example.com

Project location
/Users/an/AndroidStudioProjects/HelloAndriod

Package name
com.example.colombo.helloandriod Edit

☐ Include C++ support
☐ Include Kotlin support


Cancel Previous Next Finish



Creating an Android project




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 21: Android 5.0 (Lollipop)
- API 15: Android 4.0.3 (IceCreamSandwich)
- API 16: Android 4.1 (Jelly Bean)
- API 17: Android 4.2 (Jelly Bean)
- API 18: Android 4.3 (Jelly Bean)
- API 19: Android 4.4 (KitKat)
- API 21: Android 5.0 (Lollipop)
- API 22: Android 5.1 (Lollipop)
- API 23: Android 6.0 (Marshmallow)

☐ Wear

Minimum SDK

☐ TV

Minimum SDK

☐ Android Auto

☐ Glass

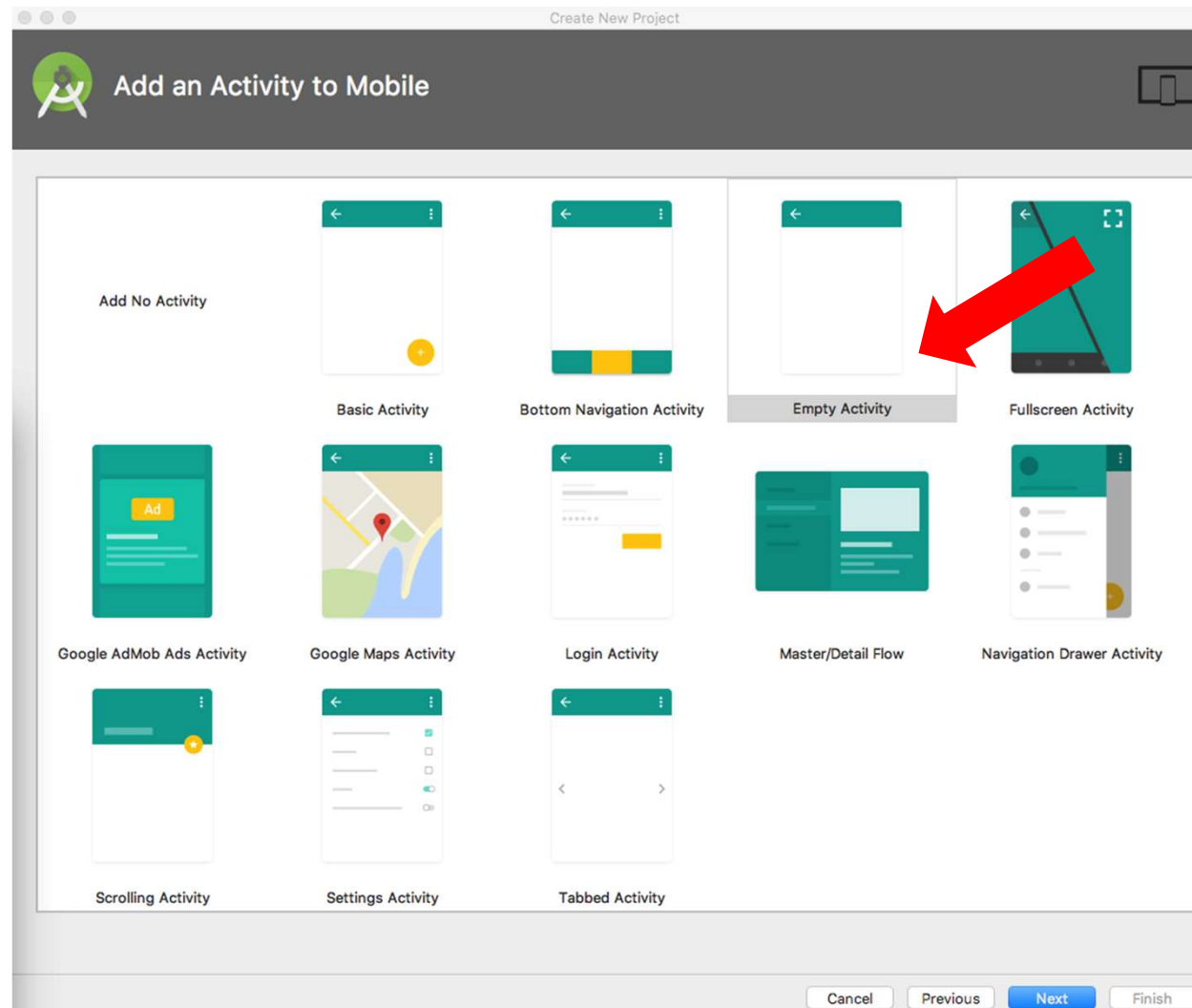
Minimum SDK

Previous Next Cancel Finish



Creating an Android project

Android Studio








Creating an Android project Android Studio



Create New Project

 **Configure Activity** 

Creates a new empty activity



Activity Name

MainActivity

☒ Generate Layout File

Layout Name

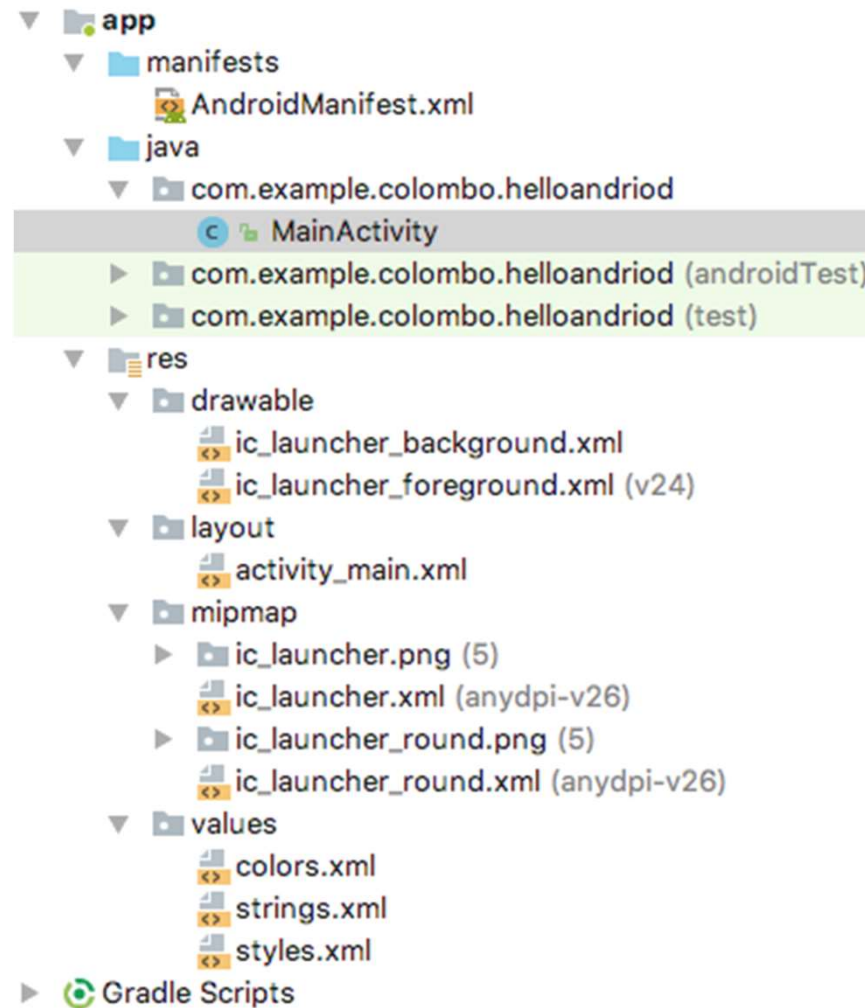
activity_main

☒ Backwards Compatibility (AppCompat)

Cancel Previous **Next** Finish

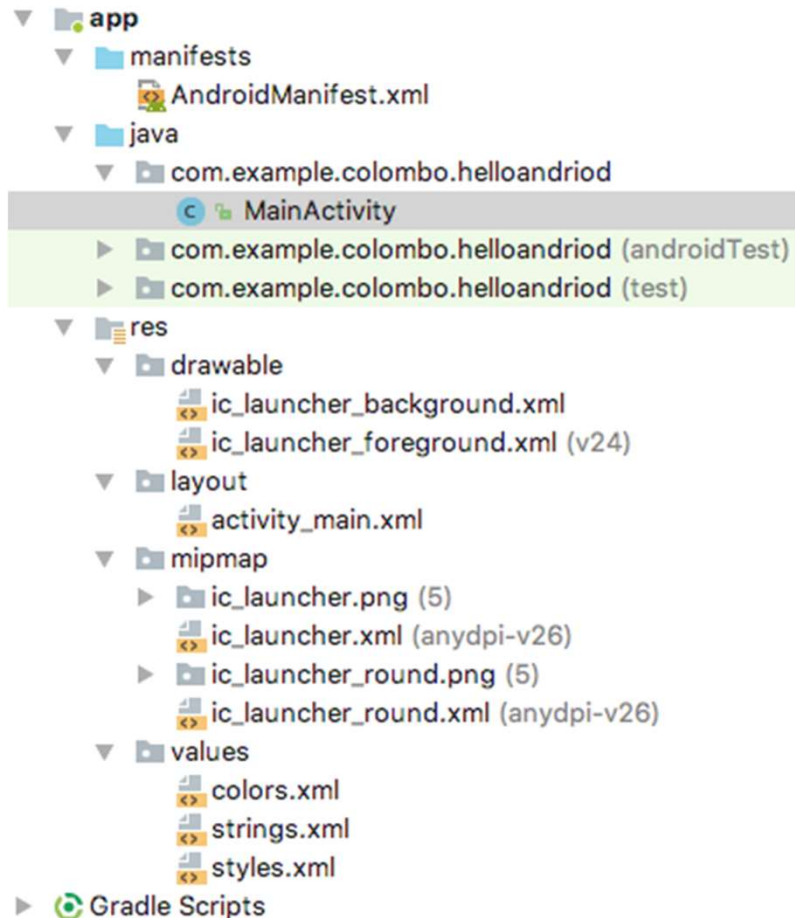


Android project files





Android project files



- **AndroidManifest.xml**—The central configuration file for the application.
- **/java folder**—Required folder for all source code.
- **/java/com..../MainActivity.java**—Main entry point to this application. This activity has been defined as the default launch activity in the Android manifest file.
- **/res folder**—Required folder where all application resources are managed. Application resources include animations, drawable graphics, layout files, data-like strings and numbers, and raw files.
- **/res/drawable-***—Application icon graphic resources are included in several sizes for different device screen resolutions.
- **/res/layout**—Layout resource file used by MainActivity to organize controls on the main application screen.
- **/res/values/strings.xml**—The resource file where string resources are defined
- **/res/values/colors.xml**
- **/res/values/dimens.xml**
- **/res/values/styles.xml**



Activity – Class MainActivity



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

public class MainActivity extends AppCompatActivity {

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

- Class *MainActivity* (default name) is located in the src directory.
- Extends *AppCompatActivity* class (in *android.support.v7.app* package)
- *AppCompatActivity*: subclass of Activity.
- An Activity provides a screen with which users can interact.
- *MainActivity*: Used as the Controller for the app.



Activity – Class MainActivity



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

public class MainActivity extends AppCompatActivity {

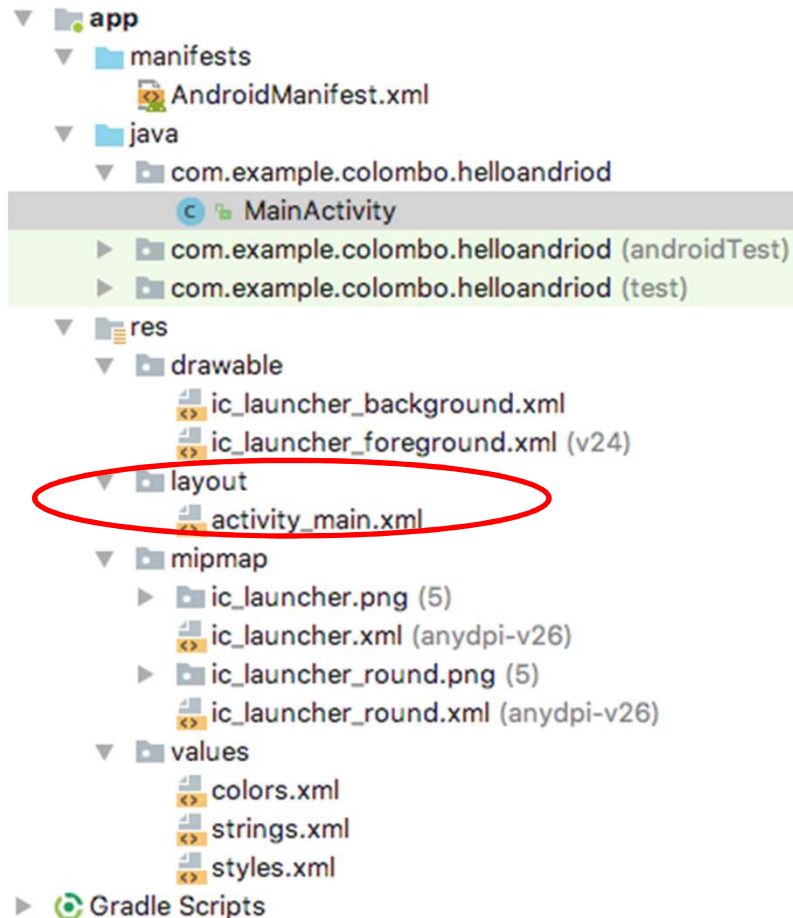
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

- *onCreate* method: called automatically.
- Inside the *onCreate* method:
 - create the initial View for the app, controlled by this Activity.
 - Inside *onCreate*, after calling the super method:
 - Set the View for this Activity by calling *setContentView* (inherited from the Activity class):
void setContentView(int layoutResID)

layoutResID is a Resource.
- The View for this Activity is set to be the one defined in the ***activity_main.xml***



activity_main.xml



- xml file, a resource, located in the layout directory of the res directory.
- automatically created when we created the project.
- defines a simple layout for this app



activity_main



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

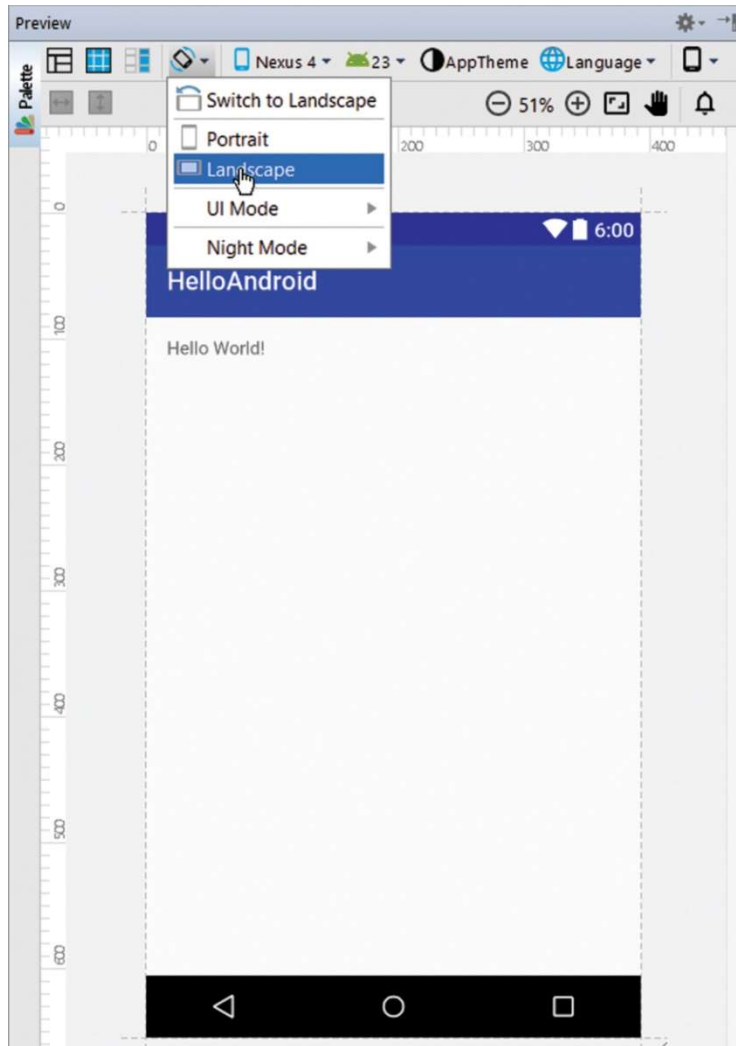
public class MainActivity extends AppCompatActivity {

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

- Access *activity_main.xml* using: ***R.layout.activity_main***
- ***activity_main*** is a static constant of the static final class layout defined inside the R class (in the R.java file): automatically created when we created the project.
- ***R*** stands for resources.



Previewing the App



- Preview a GUI defined in a layout XML file.
- View/Tool Windows/Preview
- Customise the preview:
 - Orientation
- Choose device/theme



XML (1 of 2)

- XML: eXtensible Markup Language.
- Reference: <http://www.w3.org/XML/>
- As a markup language similar to HTML, but with user-defined tags.
- A non-empty element syntax:

```
<tagName attribute1 = "value1" attribute2  
= "value2" ... >Element Content</tagName>
```

- Example:

```
<app language = "Java" version =  
"7.0">Hello Android</app>
```



XML (2 of 2)

- Element has no content (empty):

```
<tagName attribute1 = "value1" attribute2  
= "value2" ... />
```

- Ex:

```
<dev ide = "Studio" version = "1"/>
```

- Elements can be nested:

```
<a>
```

```
    <b>hello</b>
```

```
</a>
```

- Rules for naming tags: similar to naming identifiers in java.



activity_main.xml (1 of 8)

```
<?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="100dp"
    android:paddingLeft="50dp"
    android:paddingRight="50dp"
    android:paddingTop="100dp"
    tools:context="com.jblearning.helloandroid.MainActivity">

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




activity_main.xml (2 of 8)

- Two elements in activity_main.xml:
 - *RelativeLayout* and a *TextView*.
- *TextView*: nested inside the *RelativeLayout* element.
- *RelativeLayout*
 - Organizes elements in relation to each other and in relation to their parent View.



activity_main.xml (3 of 8)



```
<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout  
    ...>  
    <TextView  
        .../>  
</RelativeLayout>
```



activity_main.xml (4 of 8)

- *RelativeLayout* attributes:
 - *android:layout_width* and *android:layout_height* attributes define the size of the *RelativeLayout*.
- Value *match_parent*
 - As big as its parent element (e.g. screen).



activity_main.xml (5 of 8)

<RelativeLayout

...

android:layout_width="match_parent"

android:layout_height="match_parent"

android:paddingBottom=

"@dimen/activity_vertical_margin"

...

>



activity_main.xml (6 of 8)

- *TextView* element has three attributes.
- A *TextView* element is an instance of the *TextView* class, which encapsulates the concept of a label.

<TextView

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



activity_main.xml (7 of 8)

- Three attributes: *android:layout_width*, *android:layout_height* and *android:text*.
- *android:layout_width* and *android:layout_height* attributes define the size of the TextView.
- Value *wrap_content*: as small as possible so that their contents fit inside it.
- The element “wraps” around its content.
- By default, it appears at the top left corner of the screen.



activity_main.xml (8 of 8)

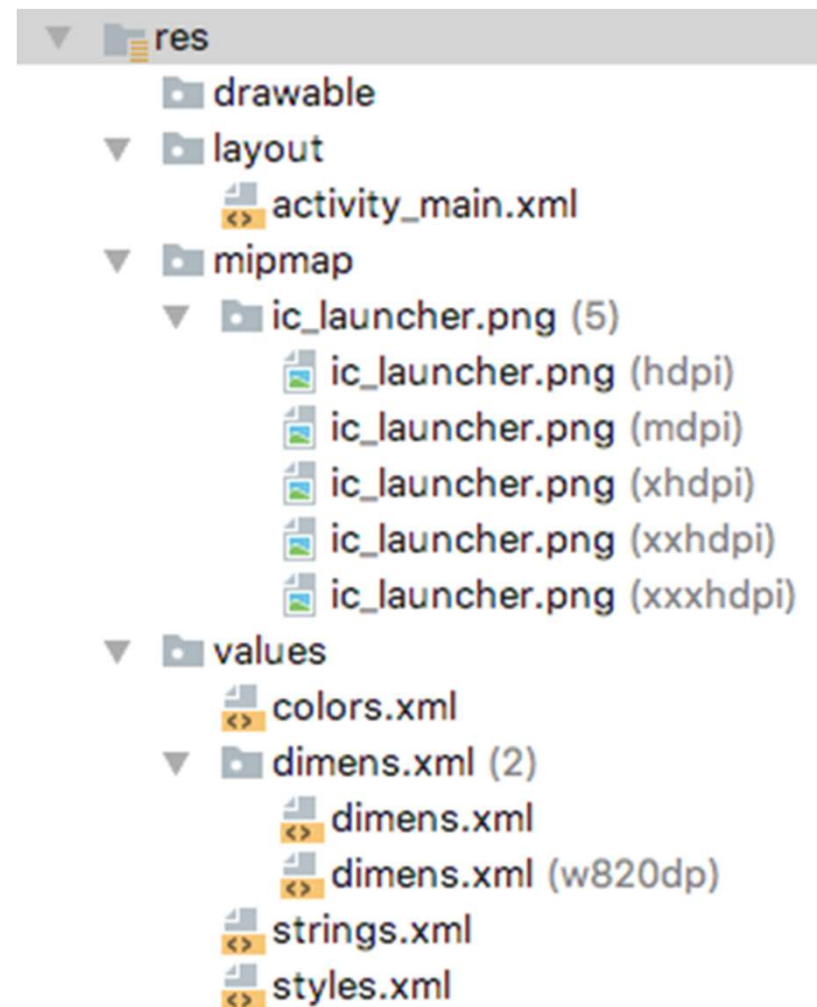
- The *android:text* attribute specifies the content of the *TextView* element.
- Its value is “Hello World!”.

```
<?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="100dp"
    android:paddingLeft="50dp"
    android:paddingRight="50dp"
    android:paddingTop="100dp"
    tools:context="com.jblearning.helloandroid.MainActivity">

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



Android Project Resources





dimens.xml (1 of 5)

- The dimens.xml file is another (automatically generated) XML file; its contents comply with XML syntax.
- It is used to define dimensions.

```
<resources>  
    <!-- Default screen margins, per the Android Design guidelines.  
    <dimen name="activity_horizontal_margin">50dp</dimen>  
    <dimen name="activity_vertical_margin">100dp</dimen>  
</resources>
```



dimens.xml (2 of 5)

```
<resources>
  <!-- Default screen margins, per the Android Design
        guidelines. -->
  <dimen
name="activity_horizontal_margin">16dp</dimen>
  <dimen name="activity_vertical_margin">16dp</dimen>
</resources>
```

- activity_horizontal_margin and activity_vertical_margin are used in activity_main.xml.



dimens.xml (3 of 5)

- The syntax for defining a dimension is
`<dimen name = "dimensionName">
valueOfDimension</dimen>`



dimens.xml (4 of 5)

- We can change the margin values as follows:

```
<resources>  
<dimen name="activity_horizontal_margin">50dp</dimen>  
  <dimen name="activity_vertical_margin">100dp</dimen>  
</resources>
```



dimens.xml (5 of 5)

- The preview shows the new margins:





strings.xml (1 of 4)

- The strings.xml is another (automatically generated) XML file; its contents comply with XML syntax.
- One string is defined in the strings.xml file: app_name.

```
<resources>  
  <string name="app_name">My First App</string>  
</resources>
```



strings.xml (2 of 4)

```
<resources>  
  <string name="app_name">HelloAndroid</string>  
</resources>
```



strings.xml (3 of 4)

- The syntax for defining a string is
`<string name =
 "stringName">valueOfString</string>`
- The value of the string `app_name` is
`HelloAndroid`



strings.xml (4 of 4)

- We can edit strings.xml, for example modify the value of hello_world as follows:
`<string name="app_name">My First App</string>`
- When we preview or run the app again, the title of the app is My First App.



activity_main.xml (1 of 2)

- We can also edit activity_main.xml, for example center the TextView as follows:

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Hi there"/>
```



activity_main.xml (2 of 2)

- When we run the app again, the text inside the TextView says “Hi there” instead of “Hello World!” as before.



styles.xml (1 of 5)

- The styles.xml file is another (automatically generated) XML file; its contents comply with XML syntax.
- It is used to define styles that the app uses.

```
<resources>

  <!-- Base application theme. -->
  <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="android:textSize">40sp</item>
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
  </style>

</resources>
```



styles.xml (2 of 5)

```
<resources>
  <!-- Base application theme. -->
  <style name="AppTheme"
    parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
  </style>
</resources>
```



styles.xml (3 of 5)

- We can modify a style by adding an *item* element using this syntax:

`<item`

`name="styleAttribute">valueOfItem</item>`



styles.xml (4 of 5)

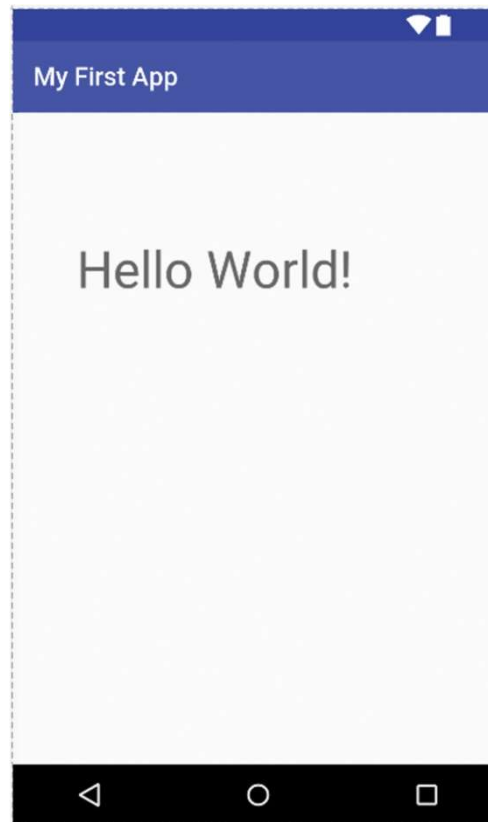
- The name of the style attribute that specifies the text size inside a TextView is android:textSize. We change the default text size to 40.

```
<item name="android:textSize">40sp</item>
```



styles.xml (5 of 5)

- The preview shows the effect of the new text size:





colors.xml (1 of 3)

- The colors.xml file is another (automatically generated) XML file; its contents comply with XML syntax.
- It is used to define colors.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="colorPrimary">#3F51B5</color>
  <color name="colorPrimaryDark">#303F9F</color>
  <color name="colorAccent">#FF4081</color>
</resources>
```



colors.xml (2 of 3)

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
  <color name="colorPrimary">#3F51B5</color>  
  <color name="colorPrimaryDark">#303F9F</color>  
  <color name="colorAccent">#FF4081</color>  
</resources>
```



colors.xml (3 of 3)

- The syntax for defining a color is
`<color name = "colorName">
 valueOfColor</color>`
- The color value is defined using hexadecimal notation (more on that later in the book).



Hexadecimal Color Value (1 of 2)



- #rrggbb (uses RGB color system).
- rr = amount of red in color
- gg = amount of green in color
- bb = amount of blue in color.



Hexadecimal Color Value (2 of 2)



- #rrggbb (uses RGB color system).
- Values vary from 00 (0) to FF (255)
- That means we can define $256 \times 256 \times 256 = 16.7$ million colors.



colors.xml (1 of 2)

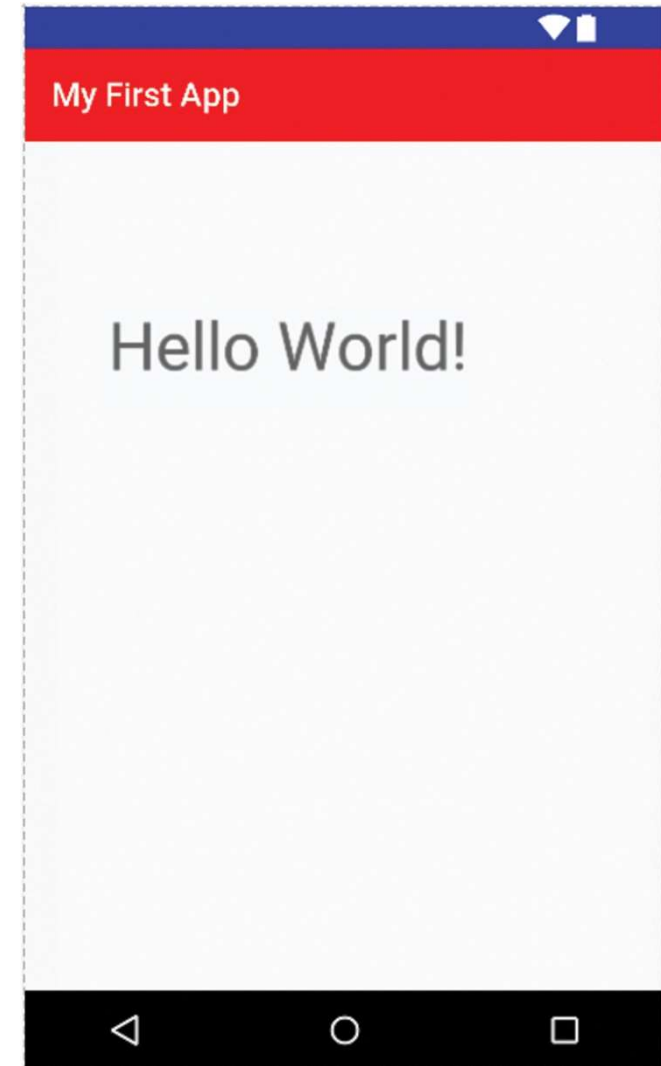
- If we modify the value of colorPrimary to red (#FF0000):

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="colorPrimary">#FF0000</color>
  <color name="colorPrimaryDark">#303F9F</color>
  <color name="colorAccent">#FF4081</color>
</resources>
```



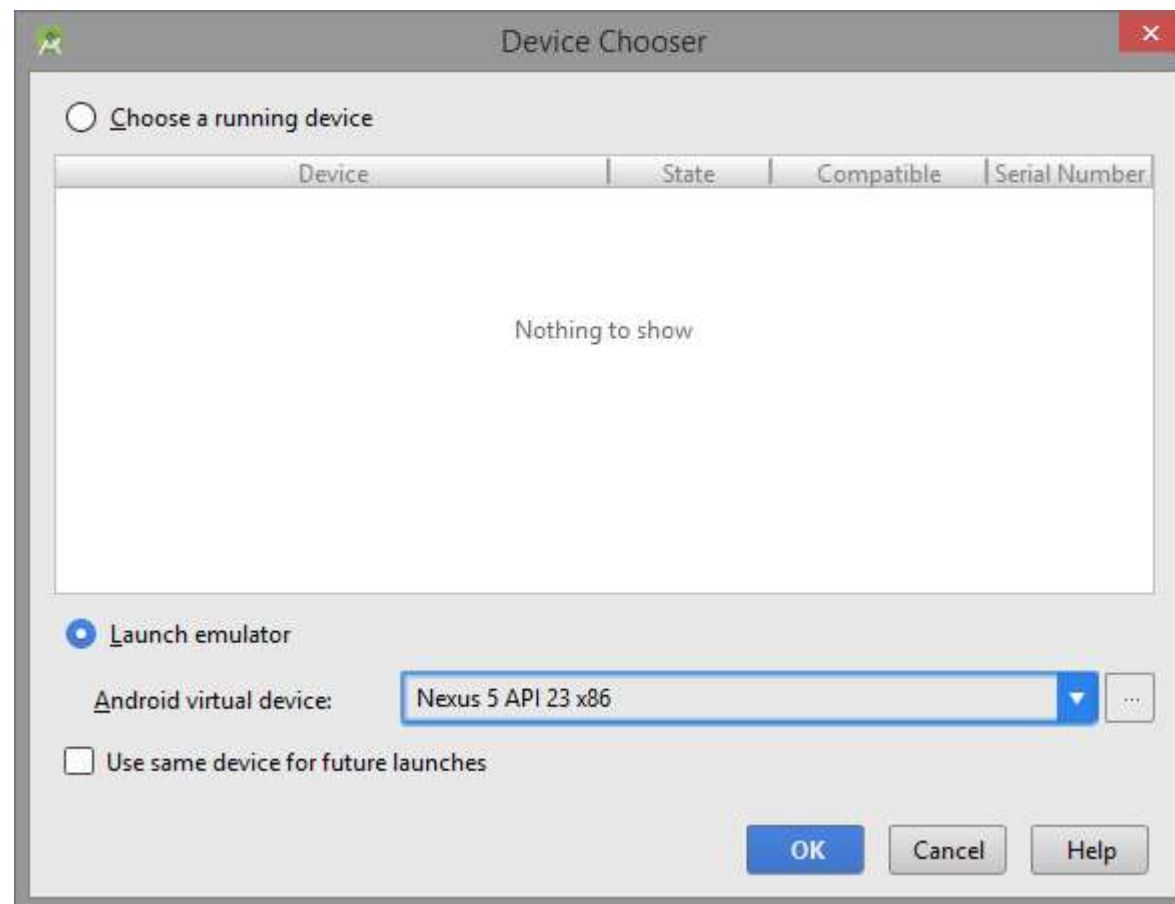
colors.xml (2 of 2)

- Previewing the app after modifying the value of color Primary to red (#FF0000):



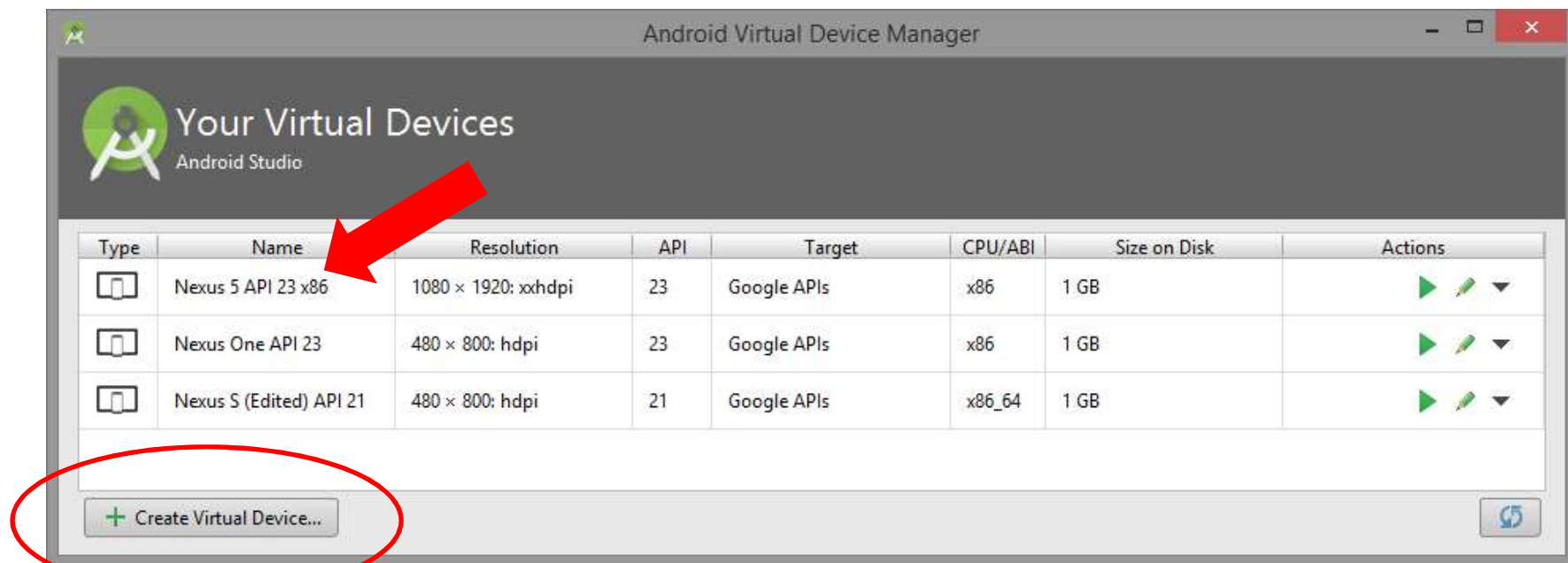


Running Android Project



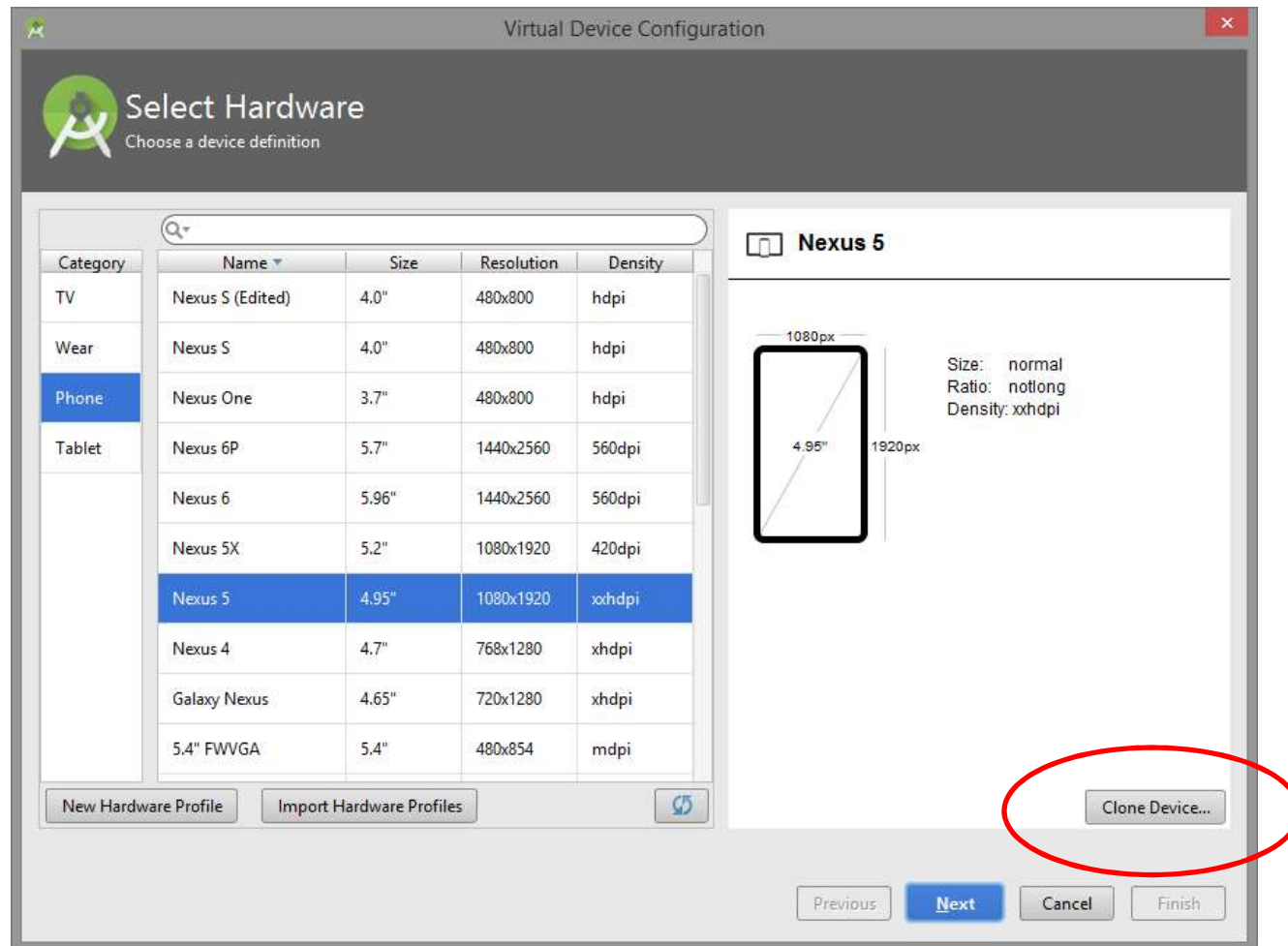


Creating Android Virtual Devices (AVD)





Creating Android Virtual Devices (AVD)






Creating Android Virtual Devices (AVD)



Hardware Profile Configuration

 Configure Hardware Profile

Device Name: Nexus 5 (Edited)

Device Type: Phone/Tablet

Screen: Screensize: 4.95 inch
Resolution: 1080 x 1920 px

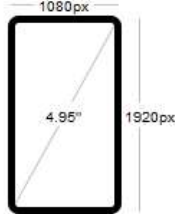
Memory: RAM: 1536 MB

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

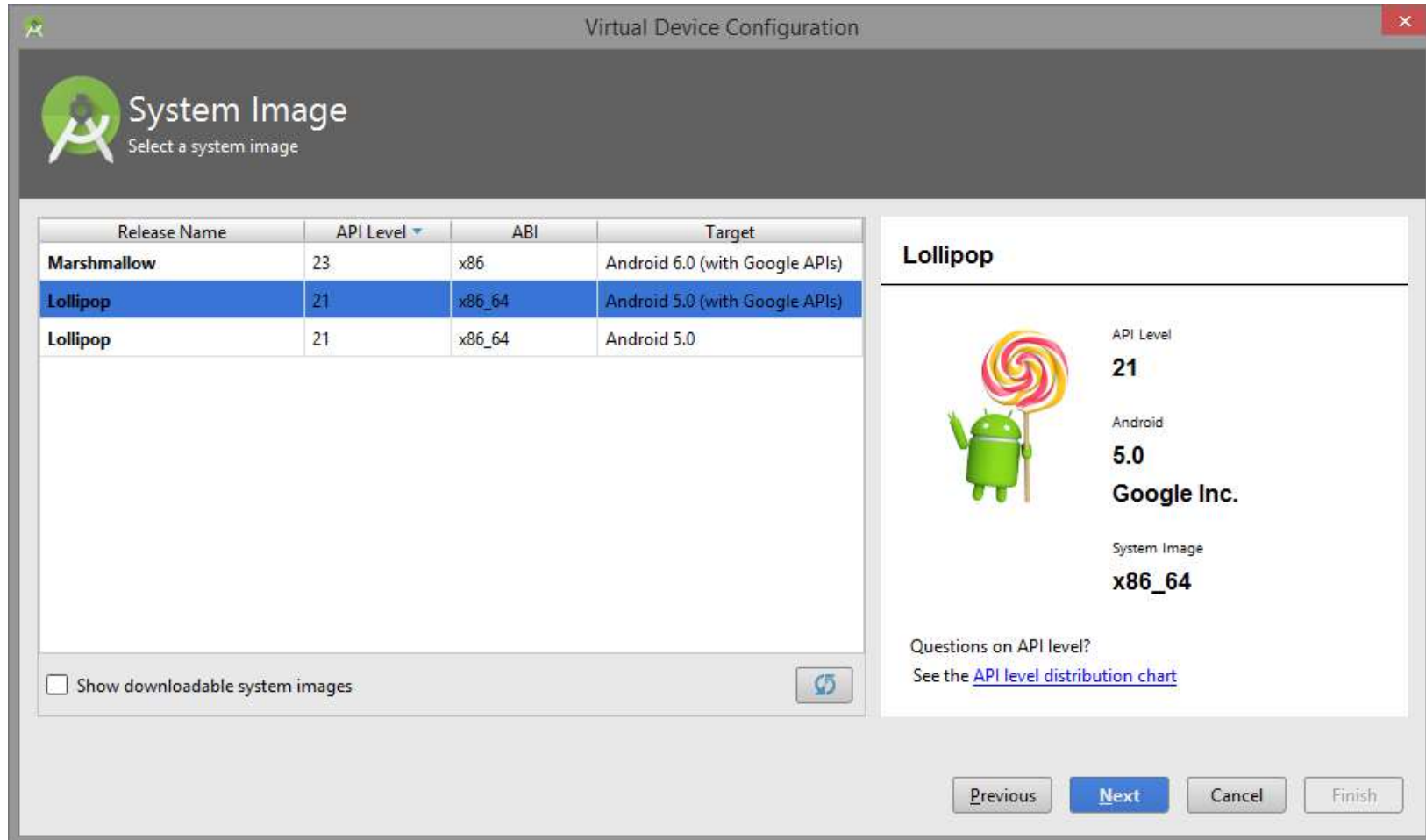
Sensors: ☒ Accelerometer
☒ Gyroscope
☒ GPS

 Size: normal
Ratio: long
Density: 420dpi

Cancel OK



Creating Android Virtual Devices (AVD)





Creating Android Virtual Devices (AVD)



Virtual Device Configuration

Android Virtual Device (AVD)
Verify Configuration

AVD Name: Nexus 5 (Edited) API 21

Nexus 5 (Edited) 4.95" 1080x1920 420dpi [Change...](#)

Lollipop Android 5.0 x86_64 [Change...](#)

Startup size and orientation

Scale: Auto

Orientation: Portrait Landscape

Emulated Performance

☒ Use Host GPU

☐ Store a snapshot for faster startup

You can either use Host GPU or Snapshots

[Show Advanced Settings](#)

Nothing Selected

[Previous](#) [Next](#) [Cancel](#) [Finish](#)



Android Device Monitor



Android Device Monitor

File Edit Run Window Help

Quick Access

DDMS

Devices

Name	Size	Date	Time	Permissions	Info
acct		2016-03-22	19:45	drwxr-xr-x	
cache		2016-02-05	23:39	drwxrwx---	
charger		1970-01-01	01:00	lrwxrwxrwx	-> /sbin/h...
config		2016-03-22	19:45	dr-x-----	
d		2016-03-22	19:45	lrwxrwxrwx	-> /sys/ker...
data		2016-03-22	19:46	drwxrwx--x	
default.pro	549	1970-01-01	01:00	-rw-r--r--	
dev		2016-03-22	19:45	drwxr-xr-x	
etc		2016-03-22	19:45	lrwxrwxrwx	-> /system...
file_contex	14591	1970-01-01	01:00	-rw-r--r--	
fstab.goldf	943	1970-01-01	01:00	-rw-r-----	
fstab.ranch	962	1970-01-01	01:00	-rw-r-----	
init	1391312	1970-01-01	01:00	-rwxr-x---	
init.envirom	852	1970-01-01	01:00	-rwxr-x---	
init.goldfis	2465	1970-01-01	01:00	-rwxr-x---	
init.ranchu	1909	1970-01-01	01:00	-rwxr-x---	
init.rc	25026	1970-01-01	01:00	-rwxr-x---	

Threads Heap Allocatio... Network ... File Expl... Emulato... System l...

LogCat

Saved Filters + -

All messages (no filter)

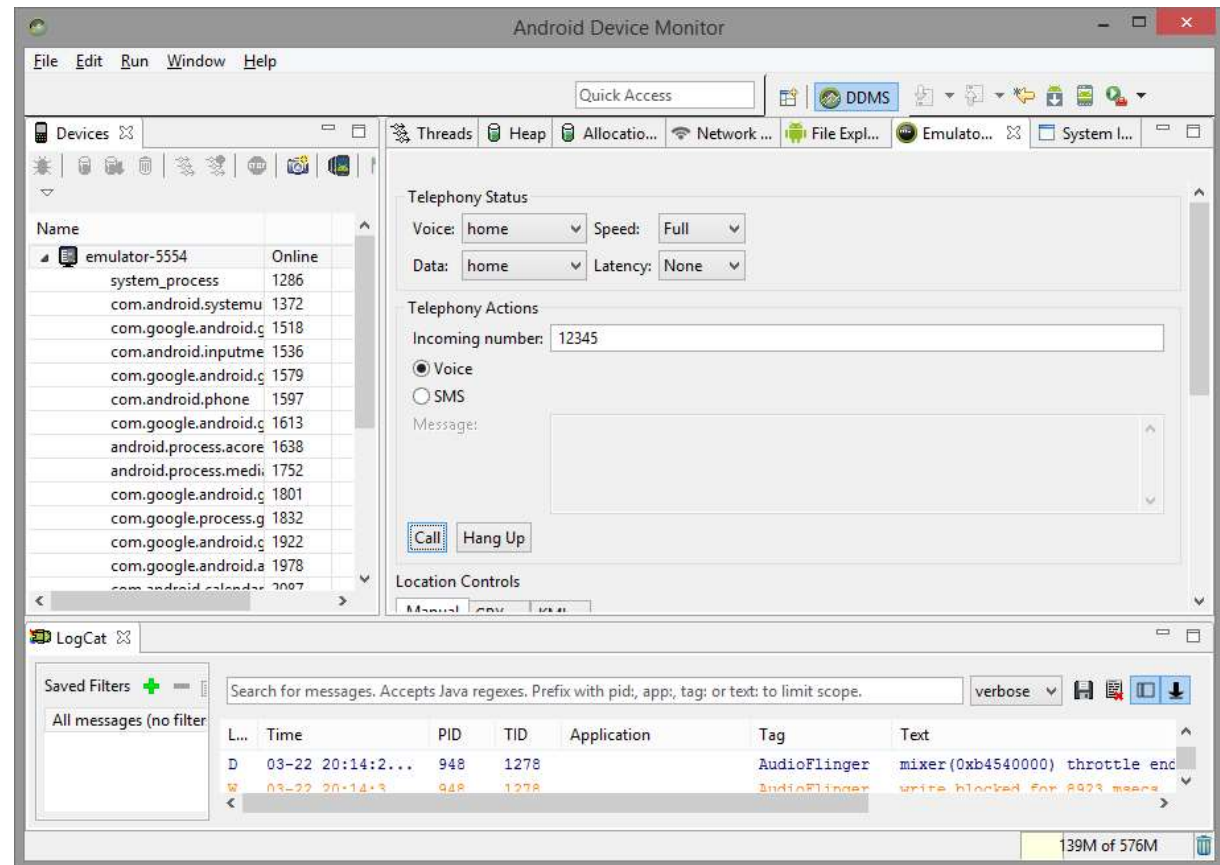
Search for messages. Accepts Java regexes. Prefix with pid, app, tag; or text: to limit scope. verbose

L...	Time	PID	TID	Application	Tag	Text
D	03-22 19:47:1...	2327	2361	com.android.cal...	InitAlarms...	Clearing and rescheduling alar
T	03-22 19:47:1...	1287	1947	system process	ActivityMa	Killing 1752:com.android.mai...

95M of 576M



Android Device Monitor









Running Inside the Emulator

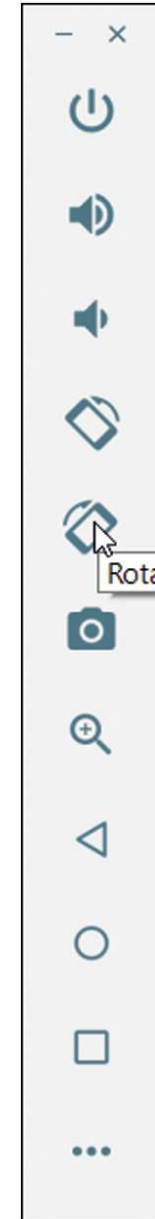




Running



- Rotate the app as it is running inside the emulator:
 - Click on the rotate button on the tool bar.

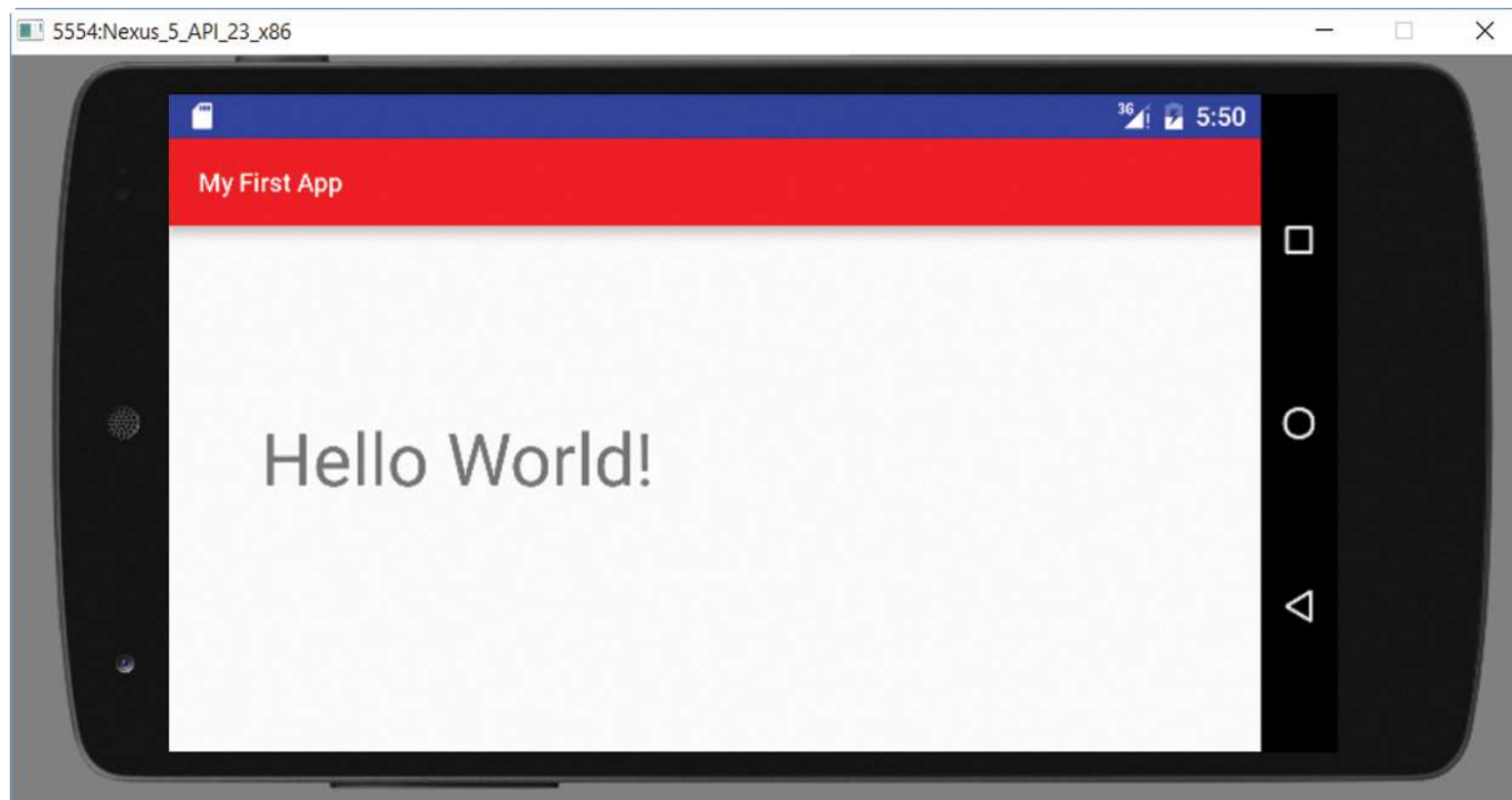




Running Inside the Emulator



- After rotating the app.





Instant Run (1 of 2)

- An interesting recent feature of Android Studio is Instant Run.
- With Instant Run, we can modify some selected components of an app, for example the strings.xml file.
- Click on the Run icon, and the emulator automatically updates the app on the fly. Instant Run is enabled by default.



Instant Run (2 of 2)

- If we want to disable Instant Run, we can choose File, Settings, and within the Build, Execution, Deployment section, select Instant Run, and then edit it.
- If we do not want to use Instant Run, we can uncheck it and check Restart activity on code changes.



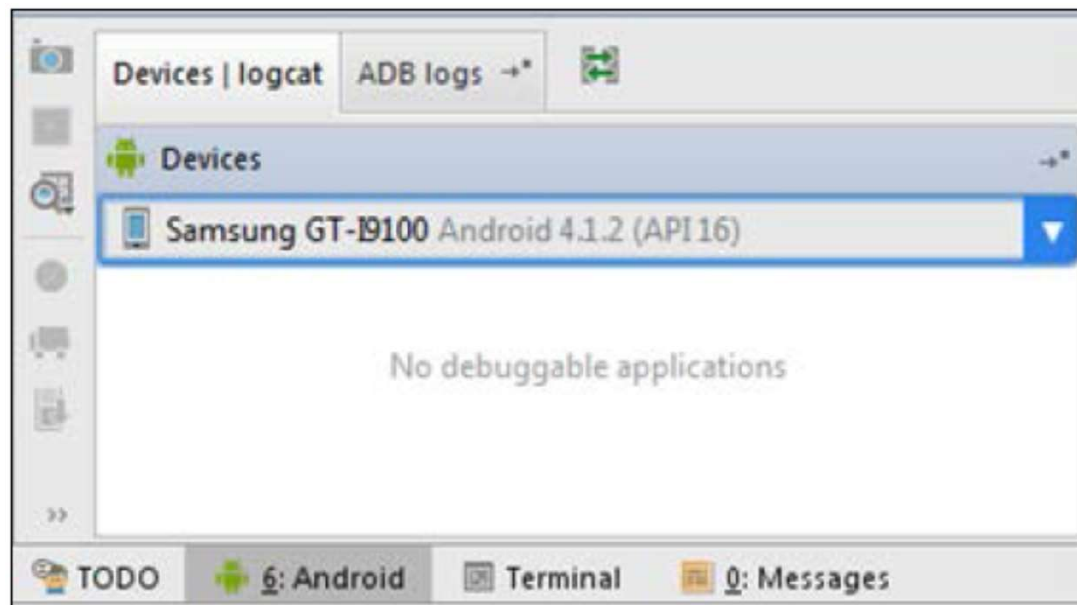
Feedback and Debugging

- We can send output to the console in addition to displaying data on the screen.
- Using various static methods of the Log class, located in the `android.util` package.



Deploy to a real Android Device Monitor

- Visit device manufacturer's website to download and install any drivers required
- <http://developer.android.com/tools/device.html>





Running on an Actual Device (1 of 5)

- To run on an actual device, we need to do two things:
 - Download a driver for the device.
 - Connect the Android device to the computer.

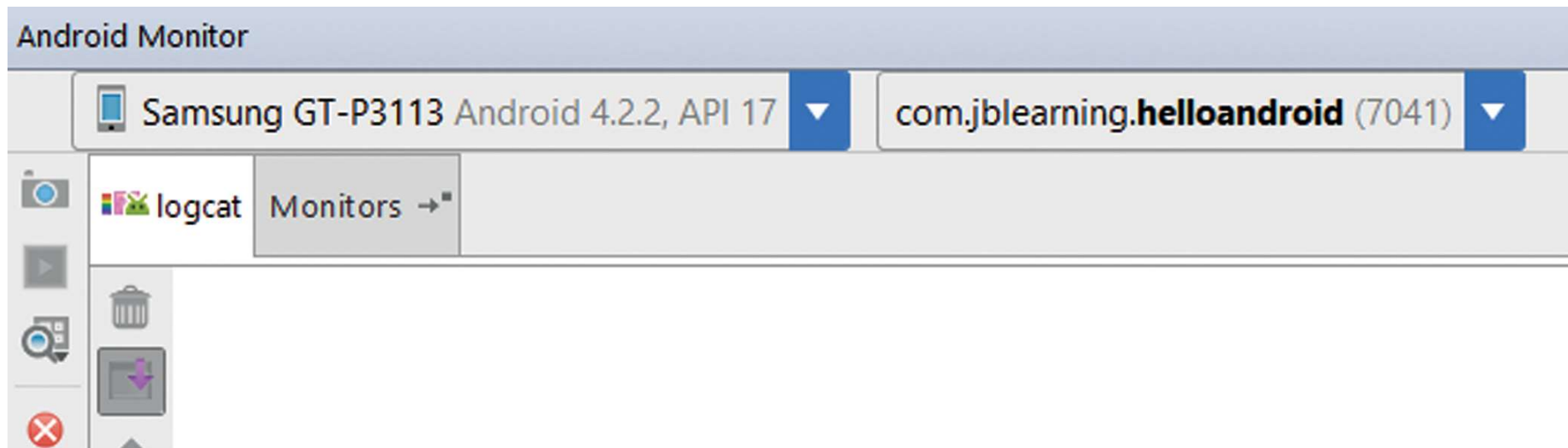


Running on an Actual Device (2



5)

- Once the device is connected, its name should appear on the left upper corner of the lower pane.

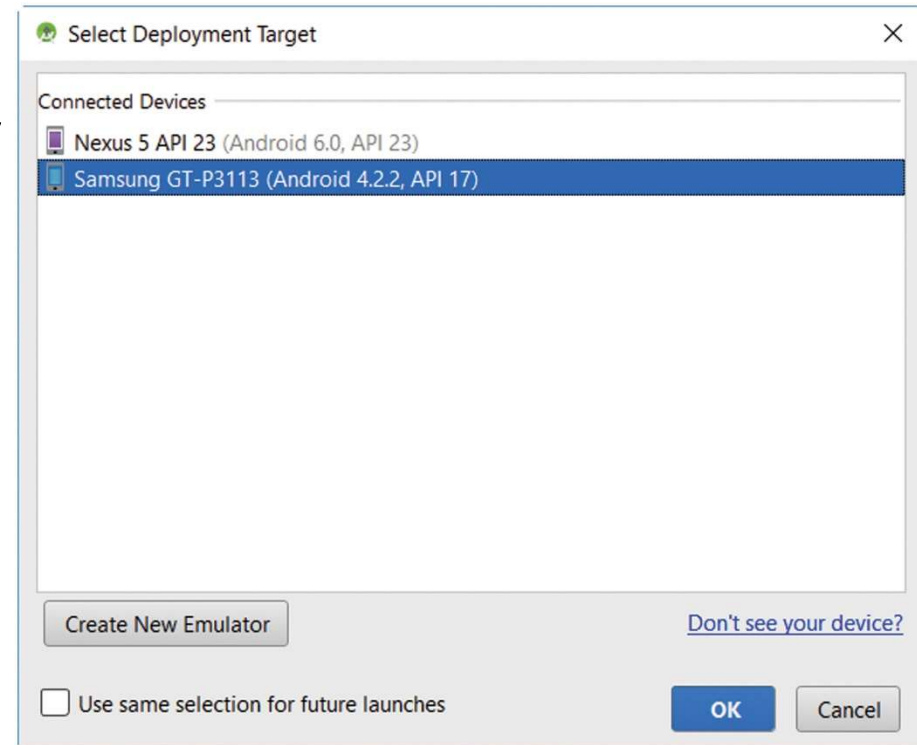




Running on an Actual Device (3 of 5)



- When we run, it should appear under "Choose a running device".
- Select "Choose a running device" and click on OK.



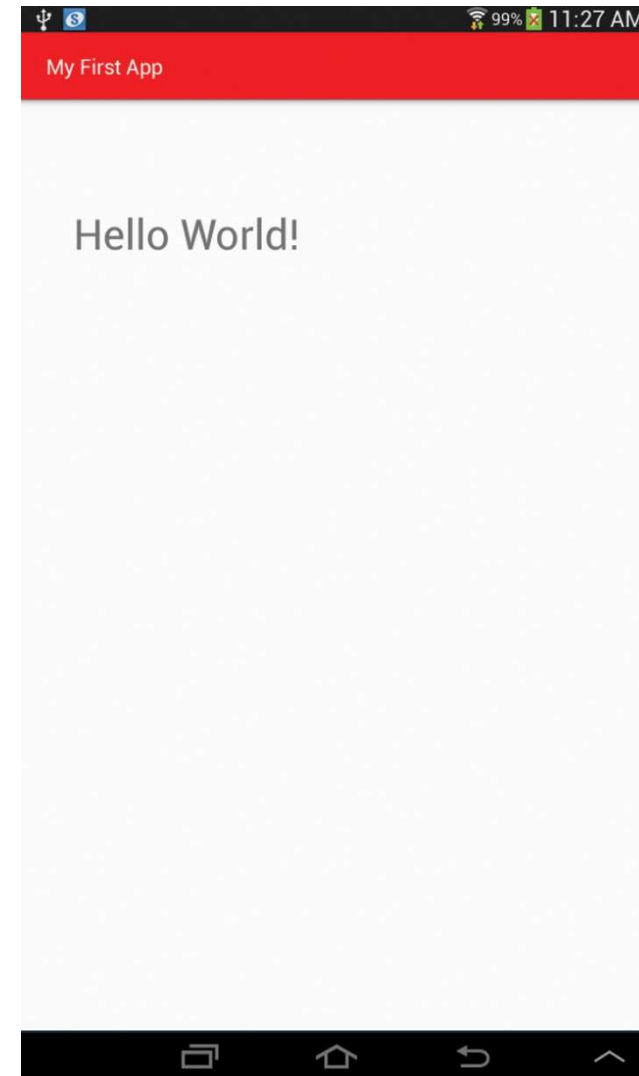


Running on an Actual Device (4



5)

- The app running inside a tablet





Running on an Actual Device (5



5)

- If the app is running on a device, we can still log output statements in Logcat.
- This is much faster than starting the emulator.



AndroidManifest.xml



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

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



The App Manifest

- The `AndroidManifest.xml` file, located in the manifests directory, specifies the resources that the app uses, such as activities, the file system, the Internet, and hardware resources.
- Before a user downloads an app on Google Play, the user is notified about these.



AndroidManifest.xml (1 of 2)

- AndroidManifest.xml's automatically generated version: Among other things, it defines the icon and the label or title for the app.
- The text inside the label is the app_name String defined in strings.xml.



AndroidManifest.xml (2 of 2)



- We should supply a launcher icon for your app.
- This is the visual representation of our app on the home screen or the apps screen.
- A launcher icon for a mobile device should be 48×48 dp.



Launcher Icon (1 of 2)

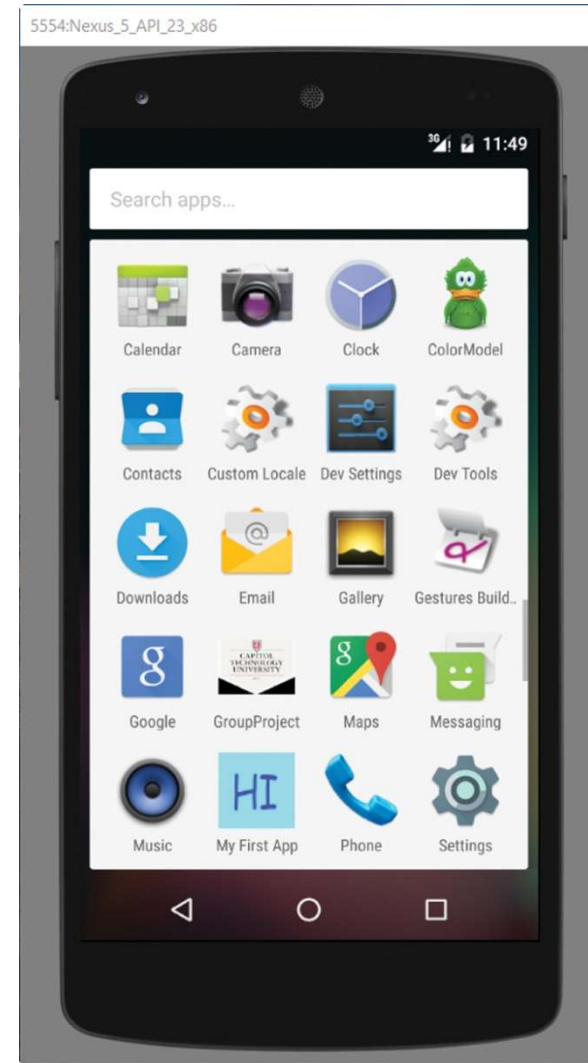
- To set the launch icon for the app to hi.png, we assign the String `@mipmap/hi` to the `android:icon` attribute of the application element in the `AndroidManifest.xml` file.
- The `@mipmap/hi` expression defines the resource in the mipmap directory (of the res directory) whose name is hi (note that we do not include the extension).

`android:icon="@mipmap/hi"`



Launcher Icon (2 of 2)

- When we run an app inside a device, the app is automatically installed.
- If we included an icon, we can see the app among the various app icons of the device.





Orientation (1 of 2)

- Sometimes, we want the app to run in only one orientation, vertical for example, and therefore we do not want the app to rotate when the user rotates the device.
- Inside the activity element, we can add the `android:screenOrientation` attribute and specify either `portrait` or `landscape` as its value.



Orientation (2 of 2)

- For example, if we want our app to run in vertical orientation only, we add:
`android:screenOrientation="portrait"`
- Note that we can control the behavior of the app on a per activity basis.
- In this app, there is only one activity, but there could be several.



Gradle Build System (1 of 2)

- Android Application Package (APK), is the file format for distributing applications that run on the Android operating system.
- The file extension is .apk.
- To create an apk file, the project is compiled and its various parts are packaged into the apk file.
- apk file can be found in the `projectName/app/build/outputs/apk` directory.



Gradle Build System (2 of 2)

- Apk files are built using the gradle build system, which is integrated in the Android Studio environment.
- When we start an app, the gradle build scripts are automatically created.
- They can be modified to build apps that require custom building.



Summary



- Android Studio
 - Creating an Android project
 - Creating a virtual device
 - Running an Android project



Summary (cont.)

- The XML layout file: `activity_main.xml`
- Other XML files: `strings.xml`, `styles.xml`, `dimens.xml`, `colors.xml`.
- Running inside the emulator
- Running on a device
- The Activity (`MainActivity`) class
- Logcat output, Debugger.
- Discussed main building blocks of Android