# Introduction to Android

Session 1

## Learning Objectives

- At the end of this meeting is expected that students will be able to:
  - Explain the Java Programming language concept on Android

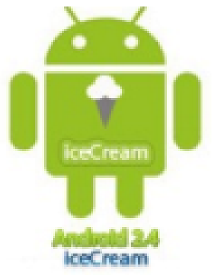
## Contents

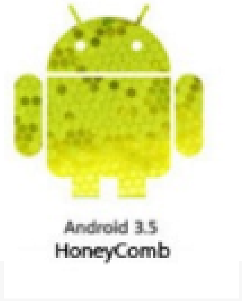
## What is Android

- Android is a **mobile operating system** that is based on a modified version of Linux.
- Developed by Android, Inc. and purchased by Google in 2005.
- It’s a powerful development framework that includes everything you need to build great apps using mix java and XML.

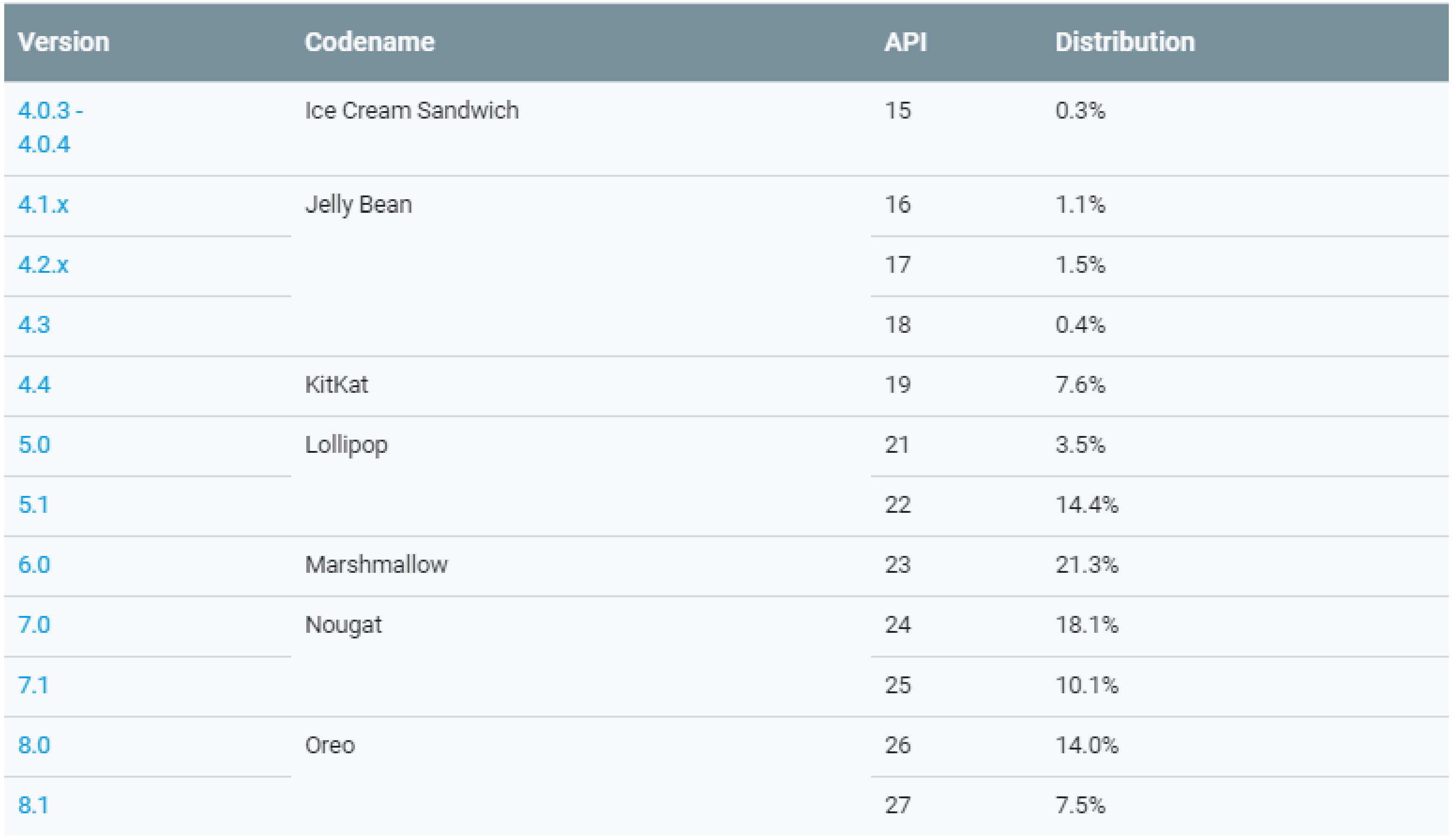
## Android Version

- **Android 1.0  -** Release Date: September 23, 2008
- **Android 1.1  -** Release Date: February 9, 2009
- **Android 1.5 Cupcake –** Release Date: April 30, 2009
- **Android 1.6 Donut -** Release Date: September 15, 2009
- **Android 2.0/2.1 Éclair –** Release Date: October 26, 2009
- **Android 2.2 Froyo (Frozen Yogurt) –** Release Date: May 20, 2010
- **Android 2.3 Gingerbread -** Release Date: December 6, 2010
- **Android 3.0 Honeycomb -** Release Date: February 22, 2011
- **Android 4.0 Ice Cream Sandwich -** Release Date: October 19, 2011
- **Android 4.1 Jelly Bean -** Release Date: July 9, 2012.
- **Android 4.4 Kit Kat** – Release Date: October 31, 2013.
- **Android 5.0 Lollipop** – Release Date: November 12, 2014.
- **Android 6.0 Marshmallow** – Release Date: October 5, 2015.
- **Android 7.0 Nougat** – Release Date: August 22, 2016.
- **Android 8.0 Oreo** – Release Date: August 21, 2017.
- **Android 9.0 Pie** – Release Date: August 6, 2018





# Android Distribution



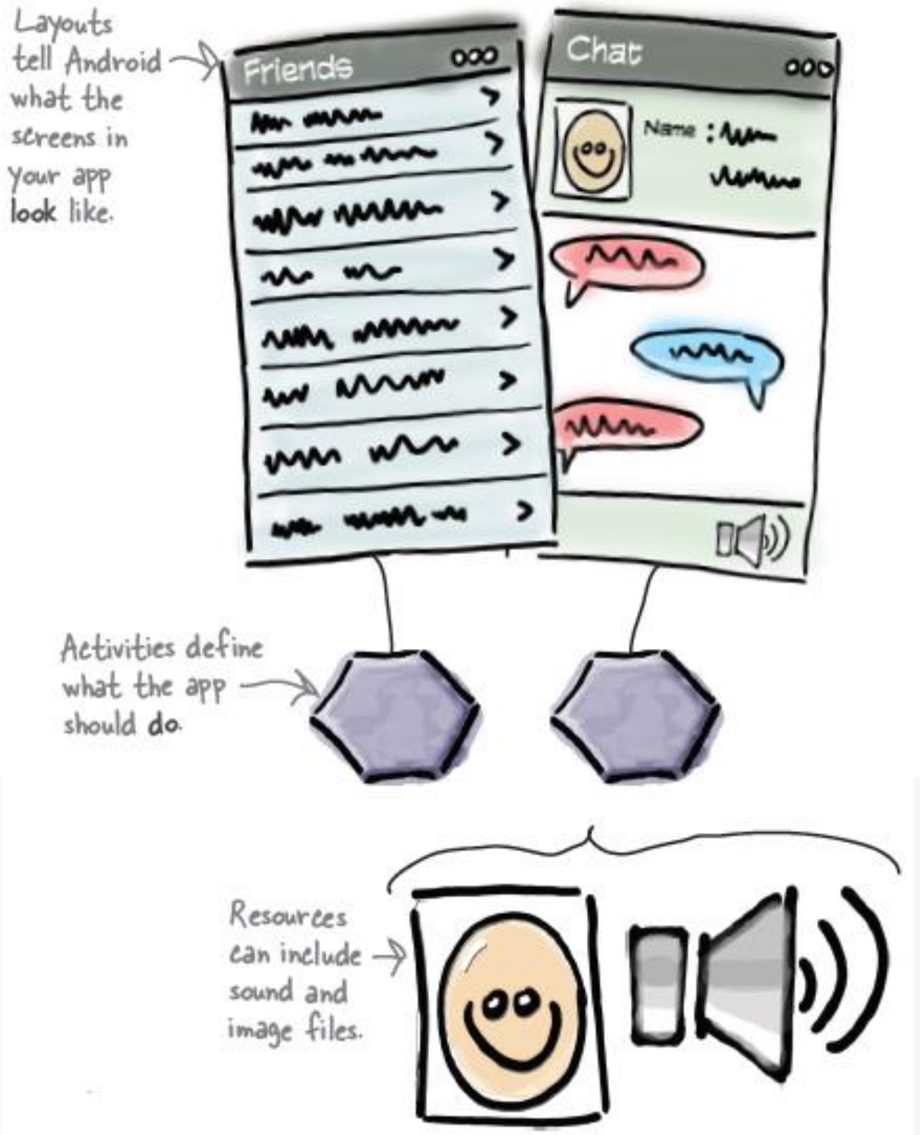
| Version | Codename | API | Distribution |
|---|---|---|---|
| 4.0.3 - 4.0.4 | Ice Cream Sandwich | 15 | 0.3% |
| 4.1.x | Jelly Bean | 16 | 1.1% |
| 4.2.x | | 17 | 1.5% |
| 4.3 | | 18 | 0.4% |
| 4.4 | KitKat | 19 | 7.6% |
| 5.0 | Lollipop | 21 | 3.5% |
| 5.1 | | 22 | 14.4% |
| 6.0 | Marshmallow | 23 | 21.3% |
| 7.0 | Nougat | 24 | 18.1% |
| 7.1 | | 25 | 10.1% |
| 8.0 | Oreo | 26 | 14.0% |
| 8.1 | | 27 | 7.5% |

## Android Architecture

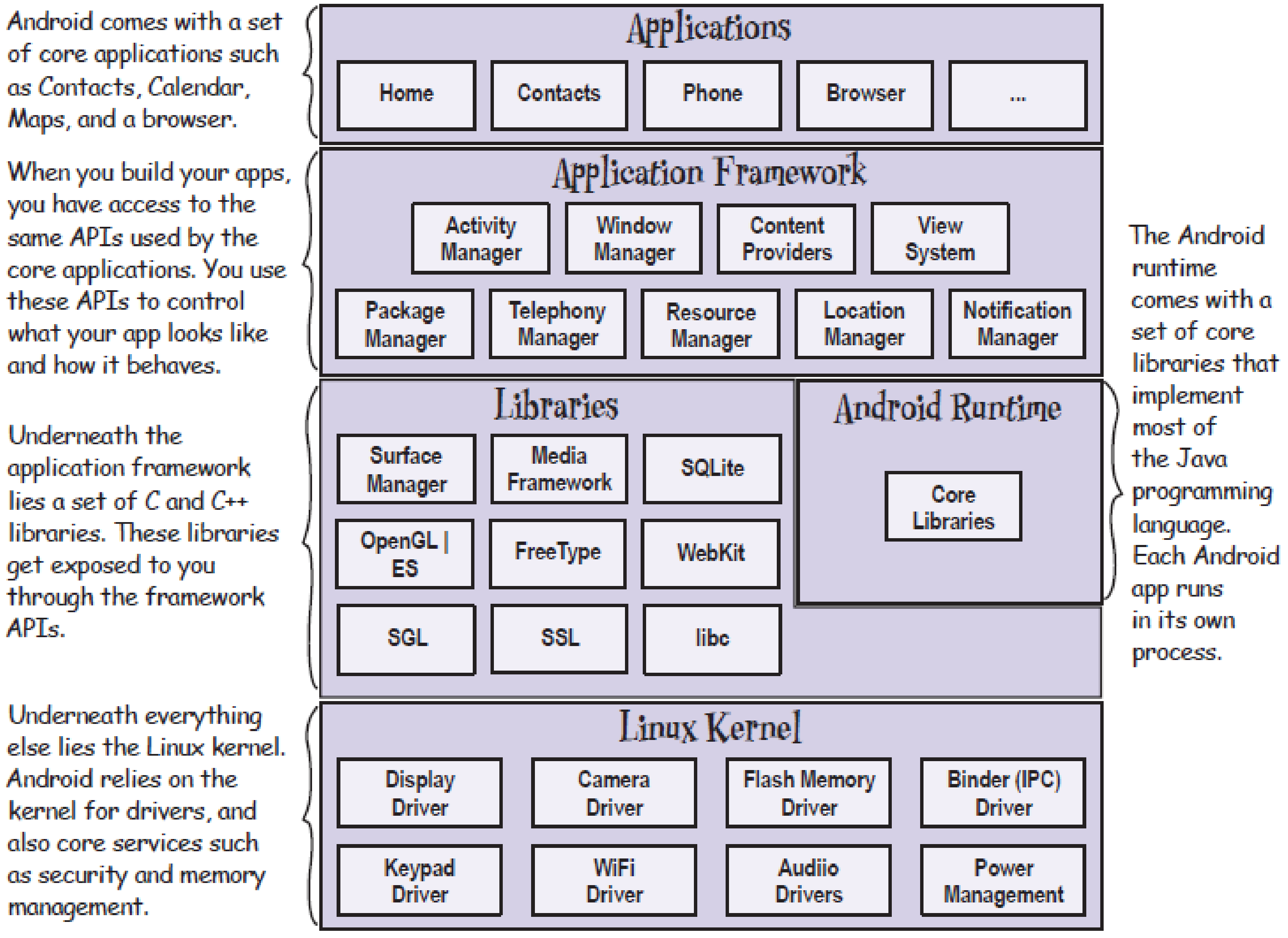
The Android OS is roughly divided into five sections in four main layers:

- Linux Kernel
- Libraries
- Android Runtime
- Application Framework
- Applications

## What makes up a typical Android App?


Layouts tell Android what the screens in your app look like.
Friends
Chat
Name :
Activities define what the app should do.
Resources can include sound and image files.

# Android Architecture



Android comes with a set of core applications such as Contacts, Calendar, Maps, and a browser.
When you build your apps, you have access to the same APIs used by the core applications. You use these APIs to control what your app looks like and how it behaves.
Underneath the application framework lies a set of C and C++ libraries. These libraries get exposed to you through the framework APIs.
Underneath everything else lies the Linux kernel. Android relies on the kernel for drivers, and also core services such as security and memory management.
The Android runtime comes with a set of core libraries that implement most of the Java programming language. Each Android app runs in its own process.
Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audiio Drivers
Power Management

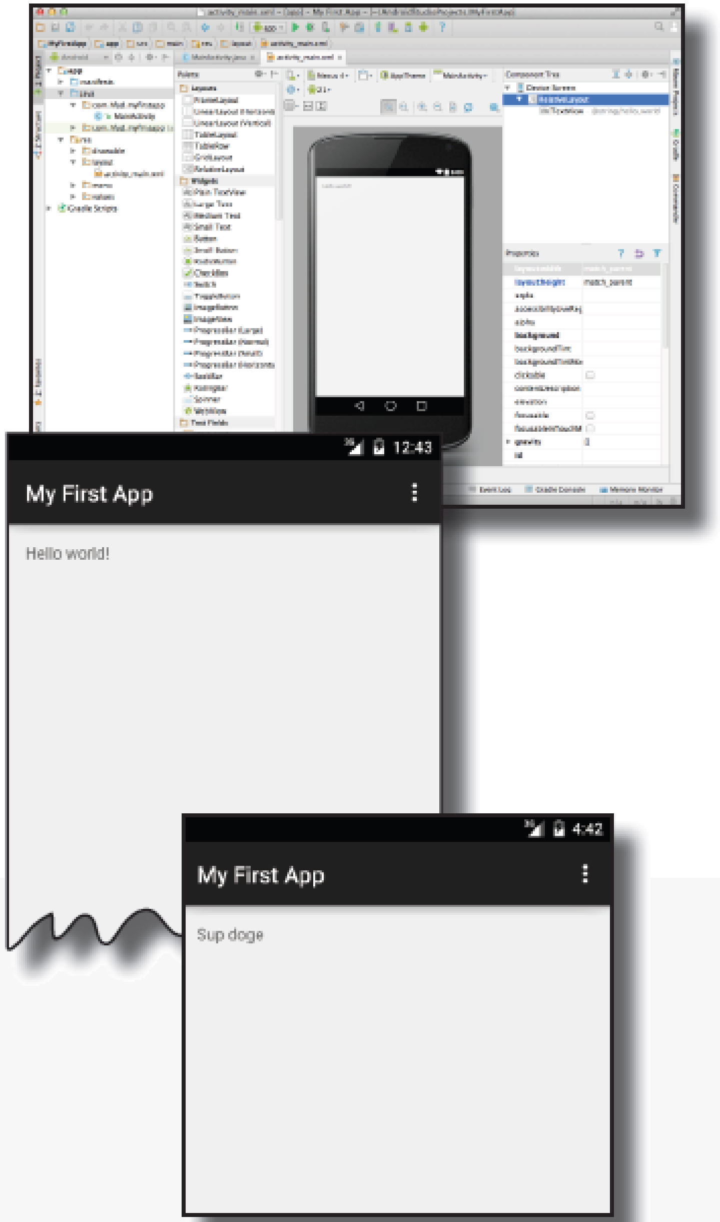
## What are we’re going to do?

1. **Set up a development environment.**
We need to install Android Studio, which includes all the tools you need to develop your Android apps.

2. **Build a basic app.**
We’ll build a simple app using Android Studio that will display some sample text on the screen.

3. **Run the app in the Android emulator.**
We’ll use the built-in emulator to see the app up and running.

4. **Change the app.**
Finally, we’ll make a few tweaks to the app we created in step 2, and run it again.



## Getting Started – Set Up Environment

### The Android SDK

**SDK Platform**
There's one of these for each version of Android.

**SDK Tools**
Tools for debugging and testing, plus other useful utilities. It also features a set of platform dependent tools.

**Sample apps**
If you want practical code examples to help you understand how to use some of the APIs, the sample apps might help you.

**Documentation**
So you can get to the latest API documentation offline.

**Android support**
Extra APIs that aren't available in the standard platform.

**Google Play Billing**
Allows you in integrate billing services in your app.

These are just some of the main points.

# Getting Started – Set Up Environment

## Android Studio is a special version of IntelliJ IDEA

- IntelliJ IDEA is one of the most popular IDEs for Java Development.
- Android Studio is a version of IDEA

## Install Java

- Android studio is Java development environment, so make sure the right java is installed on your machine.
- First, check android studio requirement of which Java Development Kit (JDK) and Java Runtime Edition (JRE).

http://developer.android.com/sdk/index.html#Requirements

- Get them and install

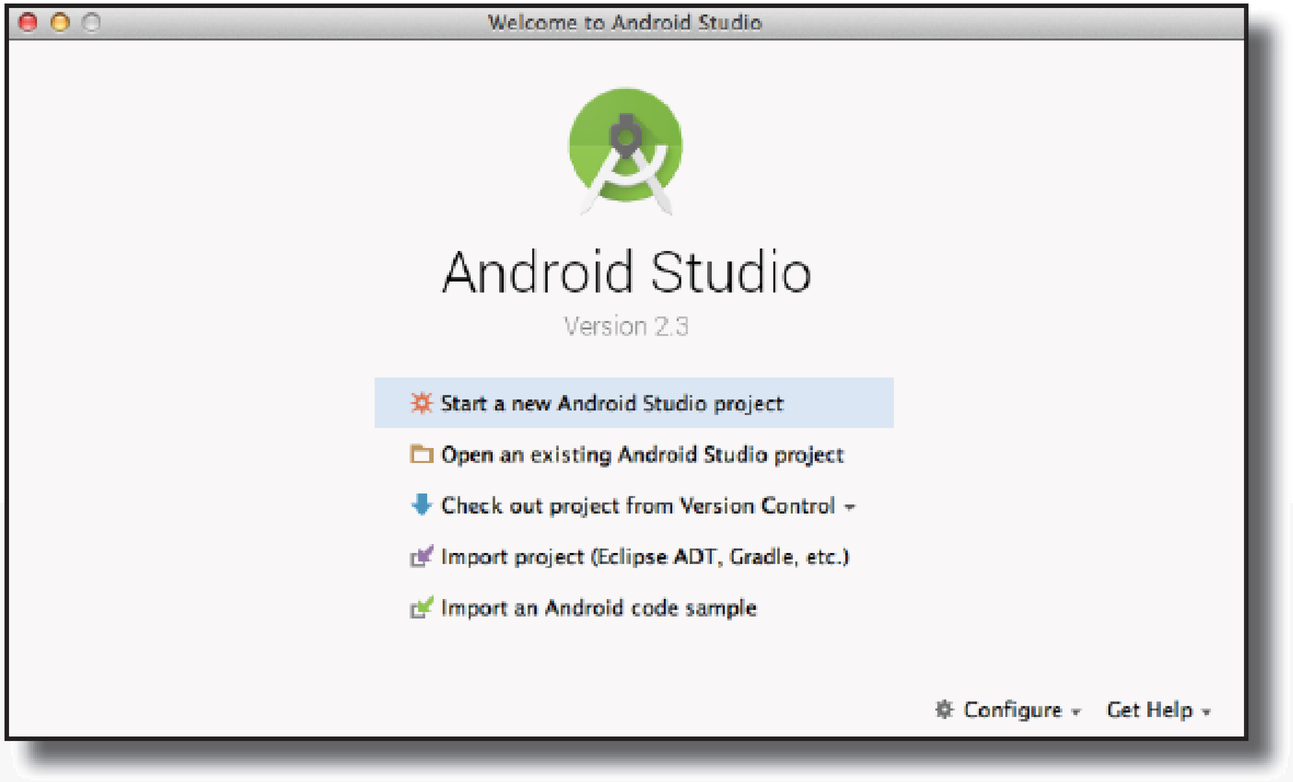
http://www.oracle.com/technetwork/java/javase/downloads/index.html

# Getting Started – Set Up Environment

## Then install Android Studio

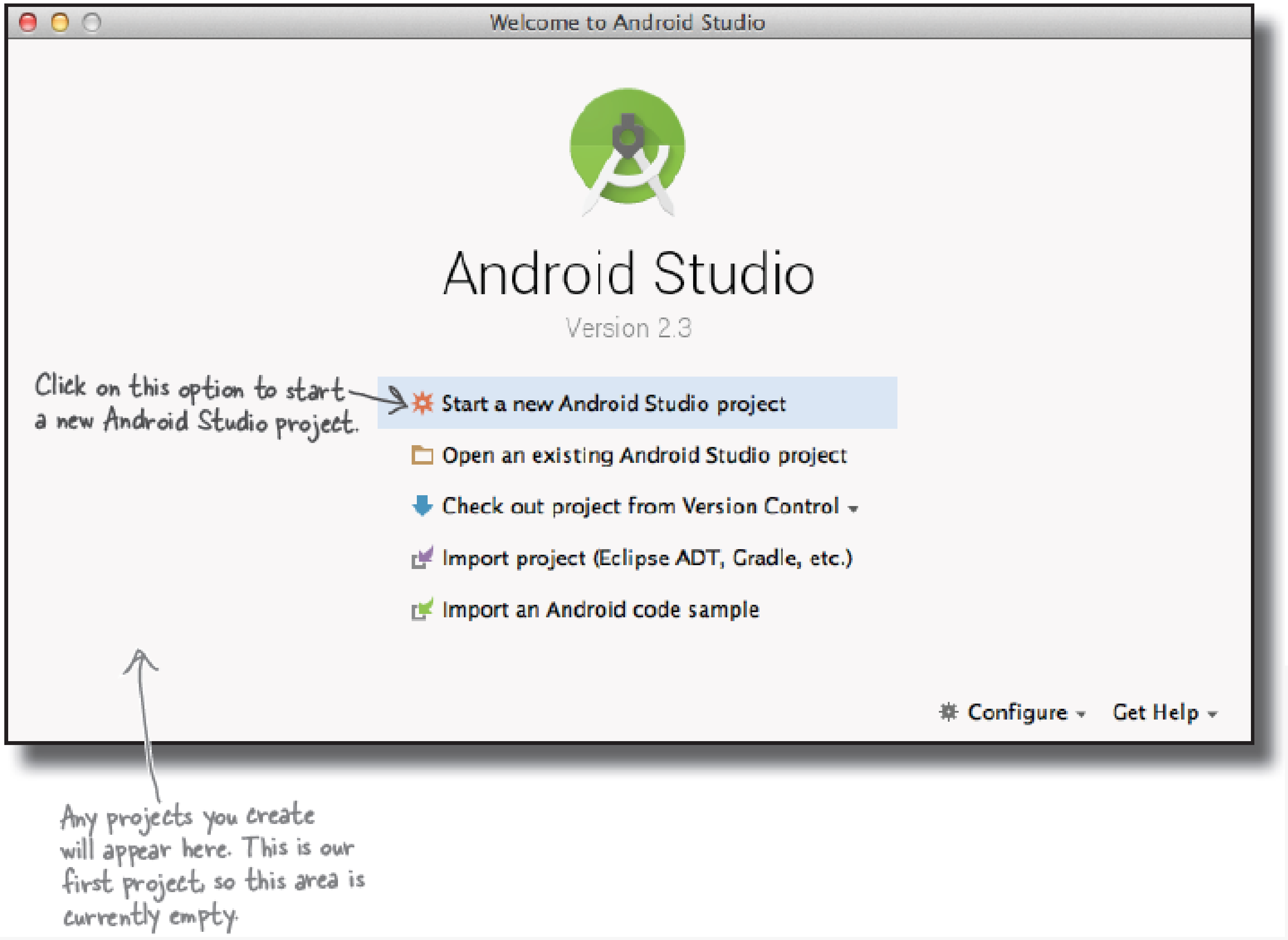
*https://developer.android.com/sdk/installing/index.html?pkg=studio*

This is the Android Studio welcome screen. It includes a set of options for things you can do.


Welcome to Android Studio
Android Studio
Version 2.3
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Import an Android code sample
Configure
Get Help

# Getting Started – Build App

## 1. Create a new Project



Welcome to Android Studio

Android Studio

Version 2.3

Click on this option to start a new Android Studio project.

Start a new Android Studio project

Open an existing Android Studio project

Check out project from Version Control

Import project (Eclipse ADT, Gradle, etc.)

Import an Android code sample

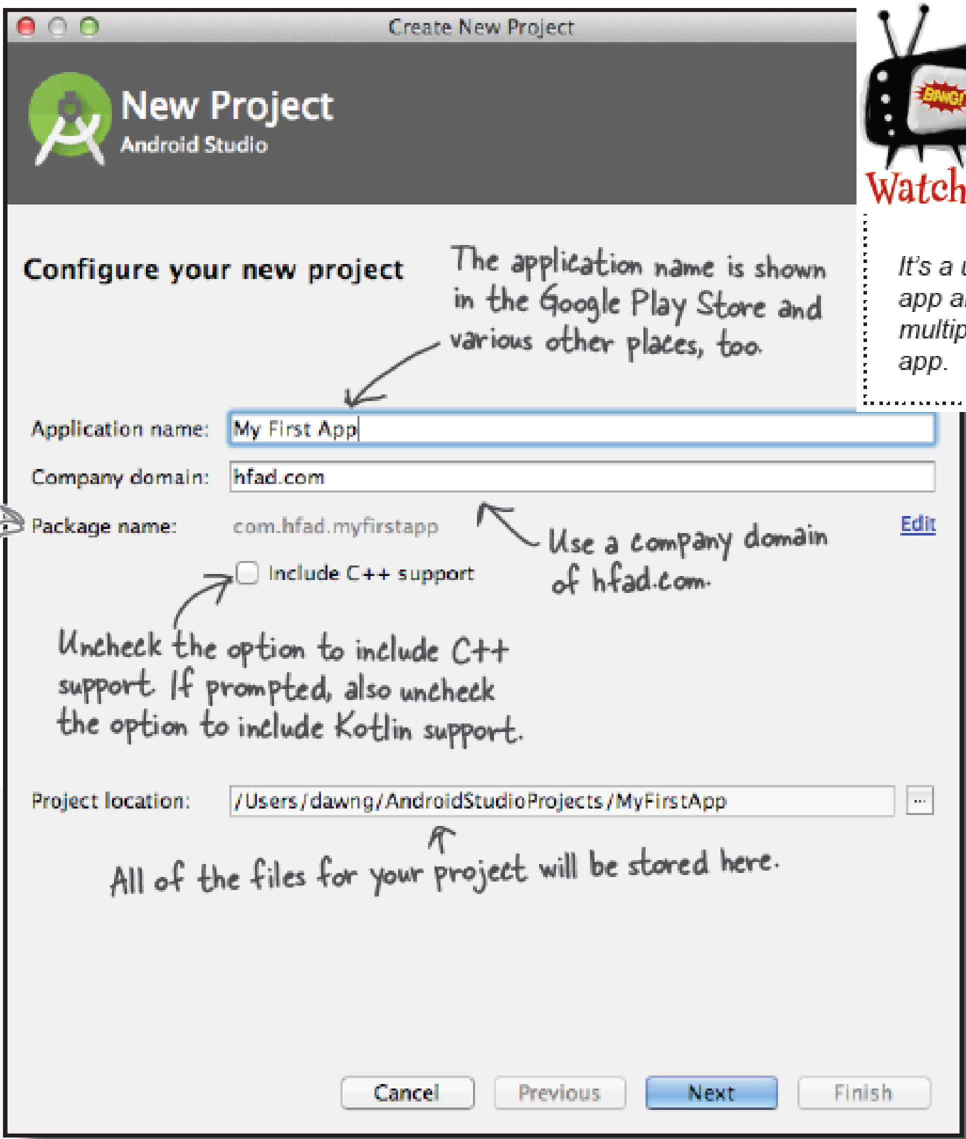
Configure

Get Help

Any projects you create will appear here. This is our first project, so this area is currently empty.

# Getting Started – Build App

## 2. Configure the Project



Create New Project

New Project
Android Studio

Configure your new project

Application name: My First App
Company domain: hfad.com
Package name: com.hfad.myfirstapp
Edit
☐ Include C++ support

Project location: /Users/dawng/AndroidStudioProjects/MyFirstApp

The application name is shown in the Google Play Store and various other places, too.

Use a company domain of hfad.com.

The wizard forms the package name by combining the application name and the company domain.

Uncheck the option to include C++ support. If prompted, also uncheck the option to include Kotlin support.

All of the files for your project will be stored here.

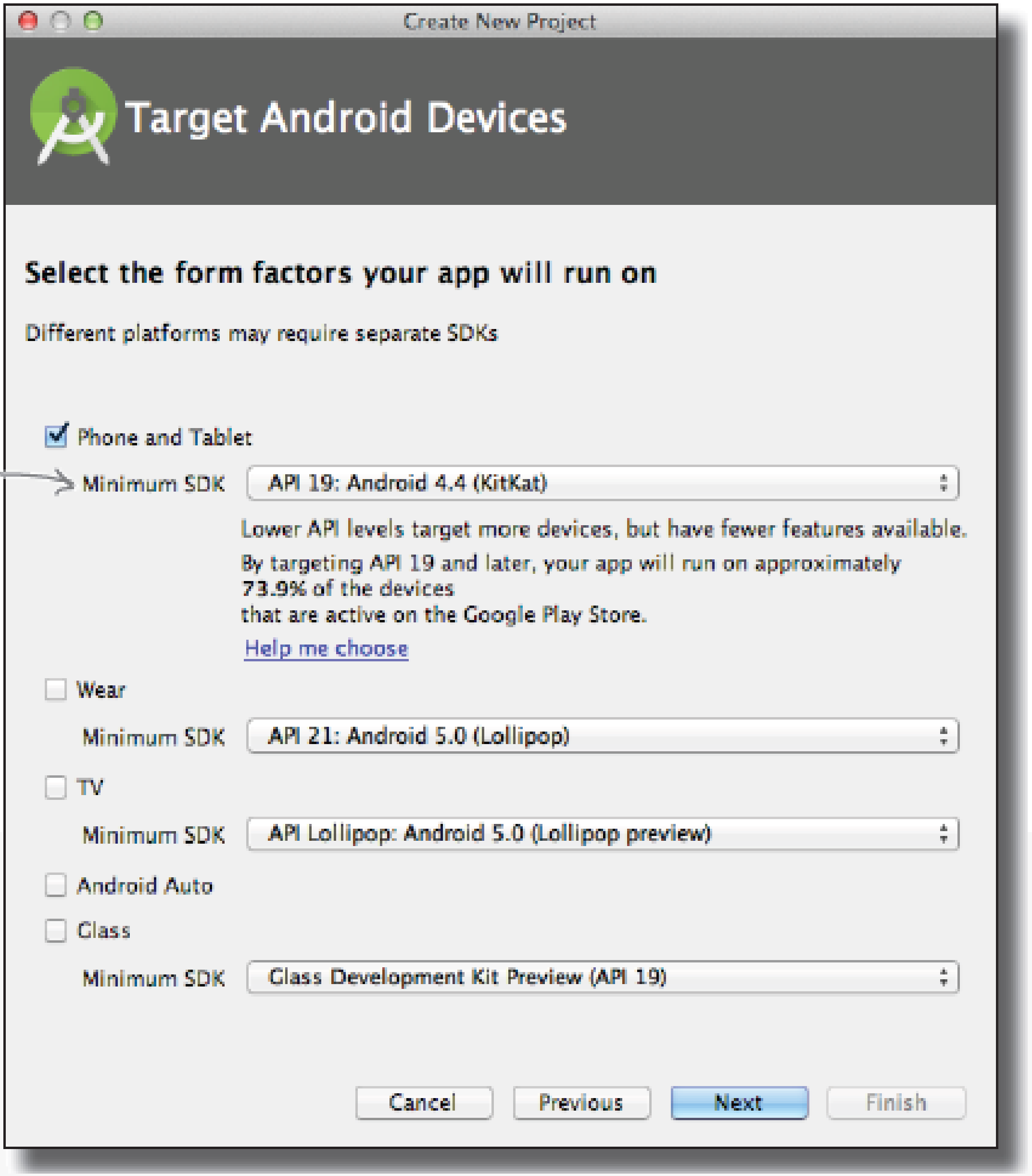
**The package name must stay the same for the lifetime of your app.**

*It's a unique identifier for your app and used to manage multiple versions of the same app.*

# Getting Started – Build App

## 3. Specify the API Level

The minimum required SDK is the lowest version your app will support. Your app will run on devices with this level API or higher. It won't run on devices with a lower API.



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 19: Android 4.4 (KitKat)

Lower API levels target more devices, but have fewer features available.

By targeting API 19 and later, your app will run on approximately 73.9% of the devices
that are active on the Google Play Store.

Help me choose

[ ] Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

[ ] TV

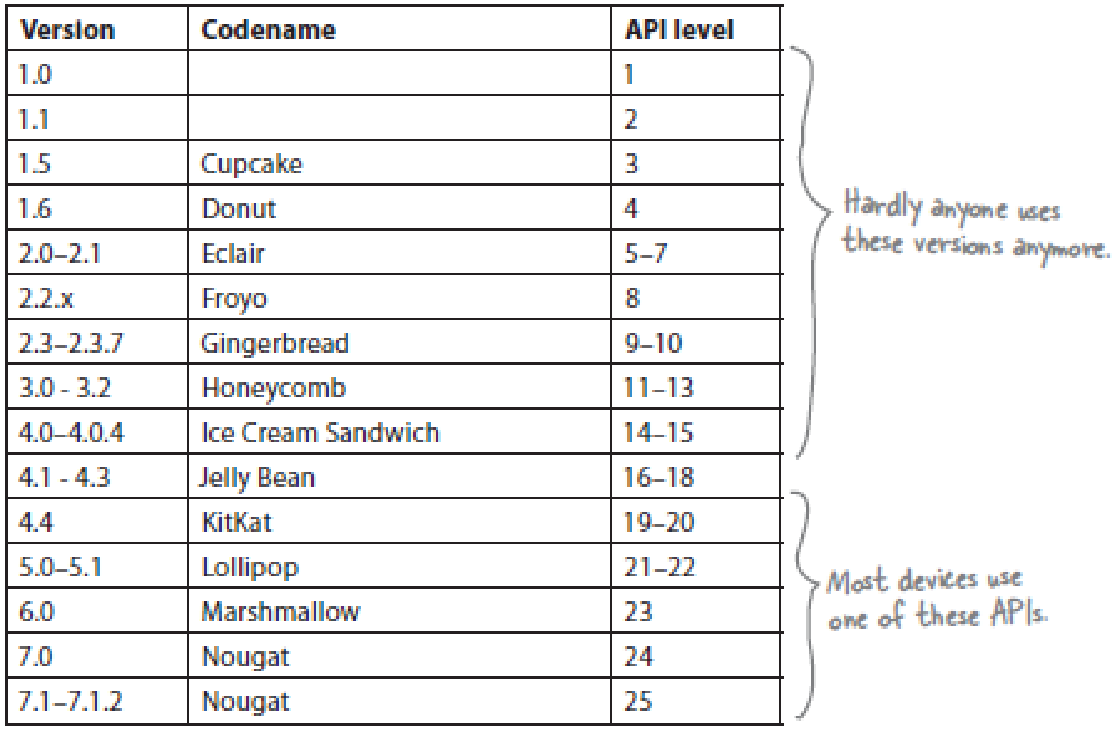
Minimum SDK: API Lollipop: Android 5.0 (Lollipop preview)

[ ] Android Auto

[ ] Glass

Minimum SDK: Glass Development Kit Preview (API 19)

# API Level for Android Version



| Version | Codename | API level |
|---|---|---|
| 1.0 | | 1 |
| 1.1 | | 2 |
| 1.5 | Cupcake | 3 |
| 1.6 | Donut | 4 |
| 2.0–2.1 | Eclair | 5–7 |
| 2.2.x | Froyo | 8 |
| 2.3–2.3.7 | Gingerbread | 9–10 |
| 3.0 - 3.2 | Honeycomb | 11–13 |
| 4.0–4.0.4 | Ice Cream Sandwich | 14–15 |
| 4.1 - 4.3 | Jelly Bean | 16–18 |
| 4.4 | KitKat | 19–20 |
| 5.0–5.1 | Lollipop | 21–22 |
| 6.0 | Marshmallow | 23 |
| 7.0 | Nougat | 24 |
| 7.1–7.1.2 | Nougat | 25 |

Hardly anyone uses these versions anymore.

Most devices use one of these APIs.

**If you specify that your app is only compatible with the very latest version of the SDK, you might find that it can't be run on many devices in the first instance.**

## Getting Started – Build App

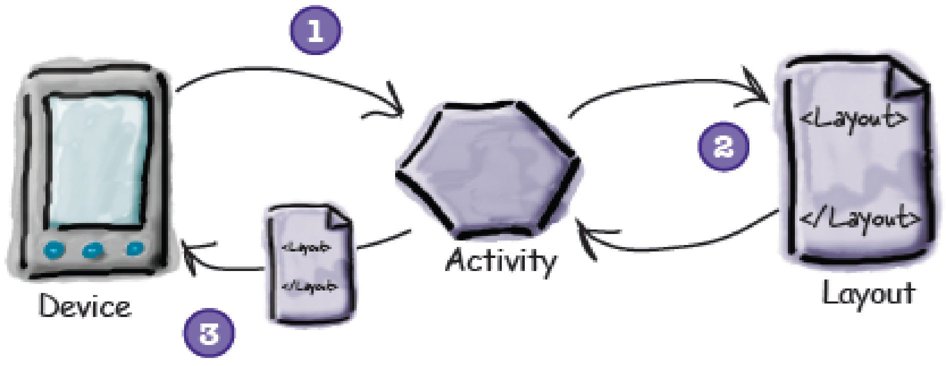
### Activity and Layout

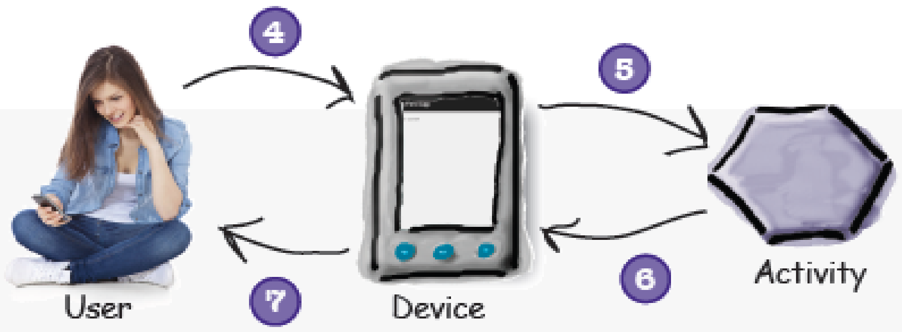
- **An activity** is a single, defined thing that your user can do. Activities are usually associated with one screen, and they’re written in java.
- **Activities** define actions.
- A **layout** describes the appearance of the screen. Layouts are written as XML files and they tell Android how the different screen elements are arranged.
- **Layout** define how the user interface is presented.

# Getting Started – Build App

## Activity and Layout

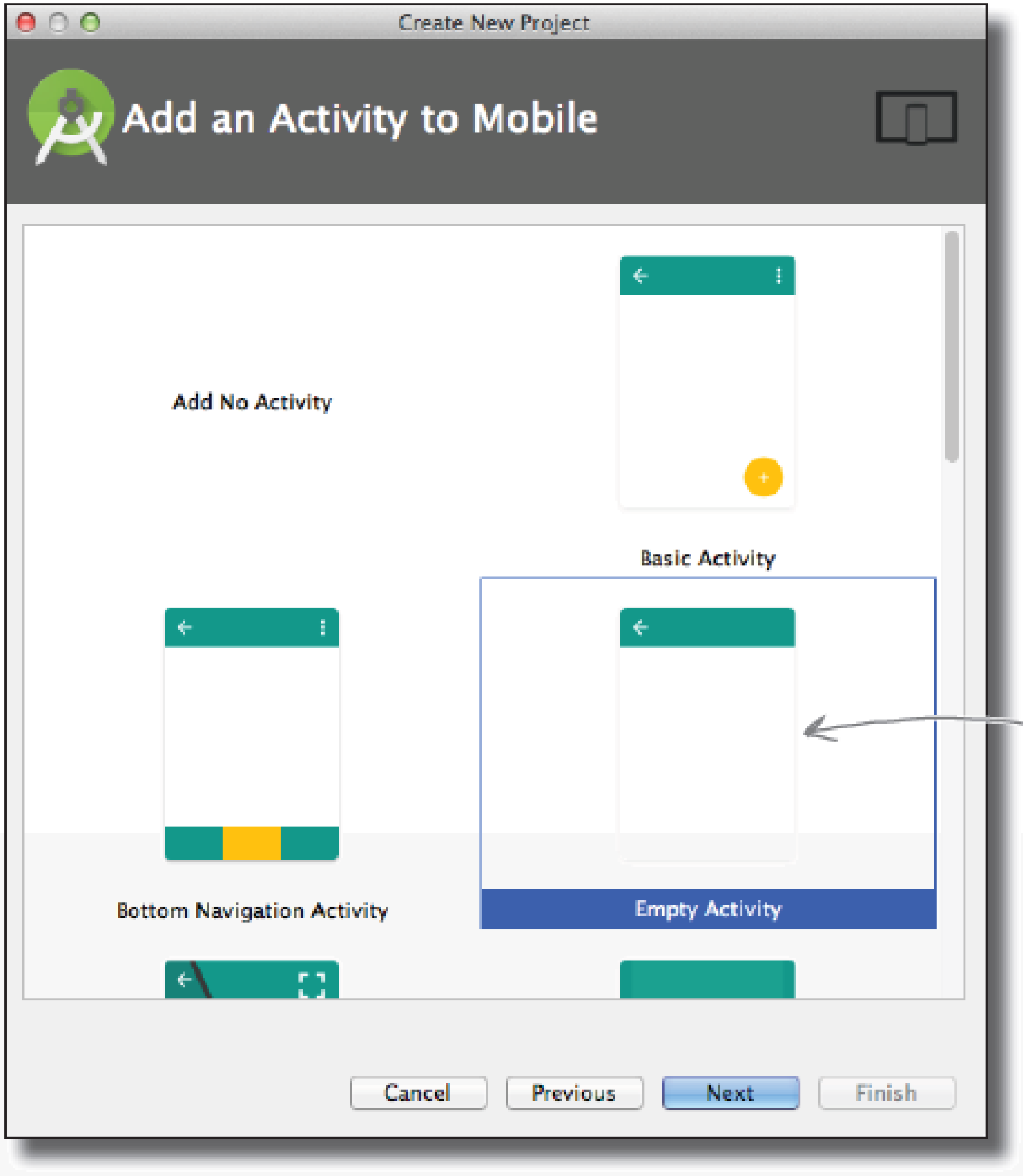
1. The device launches your app and creates an activity object.
2. The activity object specifies a layout.
3. The activity tells Android to display the layout on screen.
4. The user interacts with the layout that's displayed on the device.
5. The activity responds to these interactions by running application code.
6. The activity updates the display...
7. ...which the user sees on the device.





# Getting Started – Build App

## 4. Create an Activity

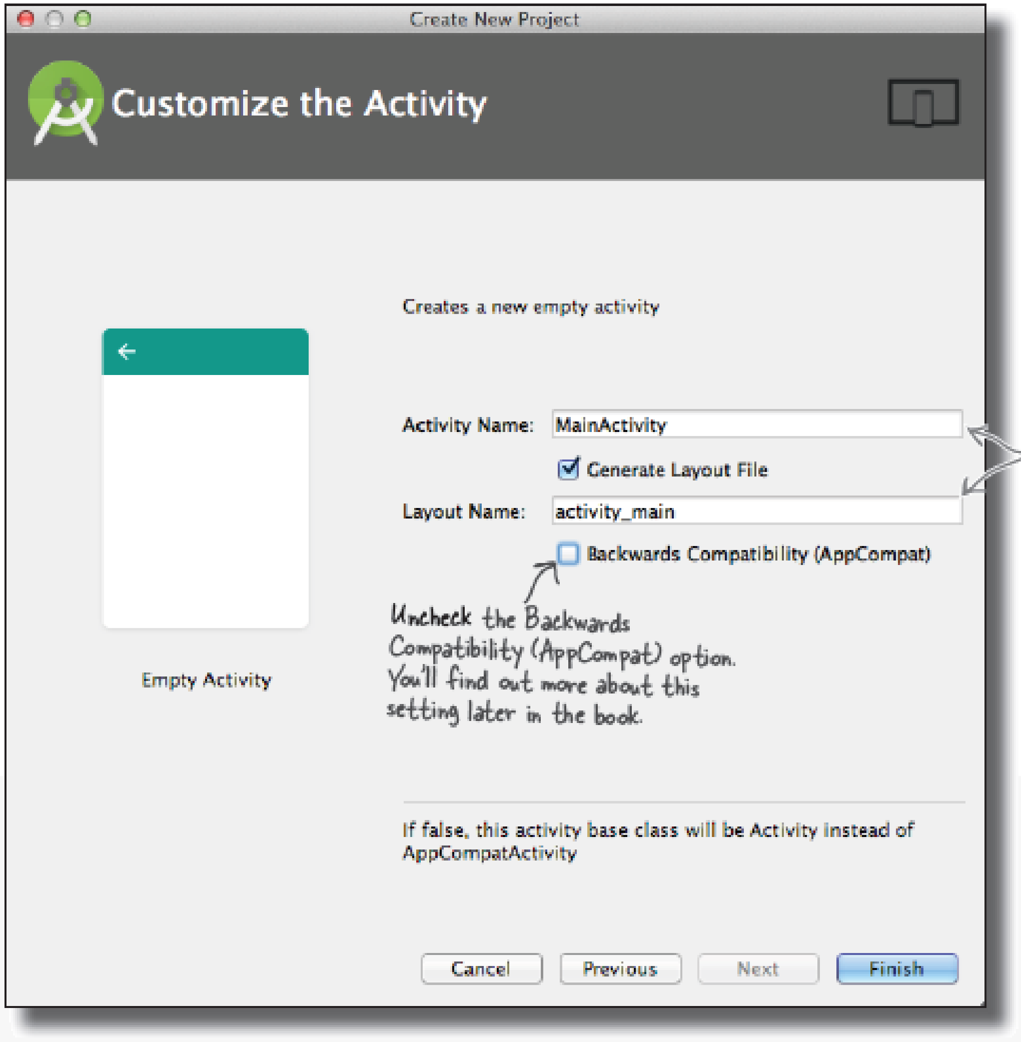


There are other types of activity you can choose from, but for this exercise make sure you select the Empty Activity option.

## Getting Started – Build App

### 5. Configure the Activity

- The activity is a Java class, and the layout is an XML file, so the names we've given here will create a Java class file called *MainActivity.java and an XML file called activity_main.xml*
.



Create New Project

Customize the Activity

Creates a new empty activity

Empty Activity

Activity Name: MainActivity

[✓] Generate Layout File

Layout Name: activity_main

[ ] Backwards Compatibility (AppCompat)

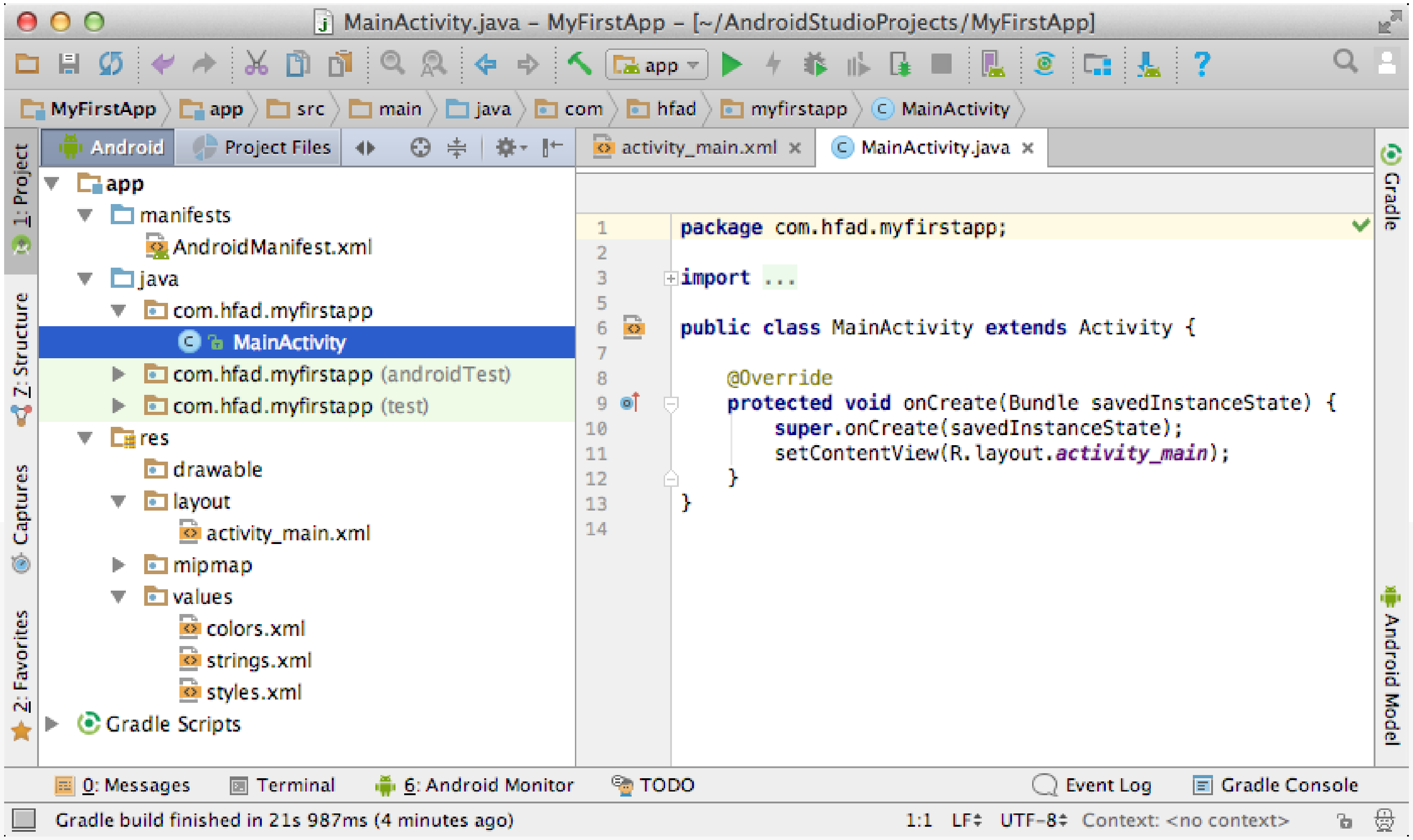
Uncheck the Backwards Compatibility (AppCompat) option. You'll find out more about this setting later in the book.

If false, this activity base class will be Activity instead of AppCompatActivity

Name the activity "MainActivity" and the layout "activity_main". Also make sure the option to generate the layout is checked.

# What Just Happened?

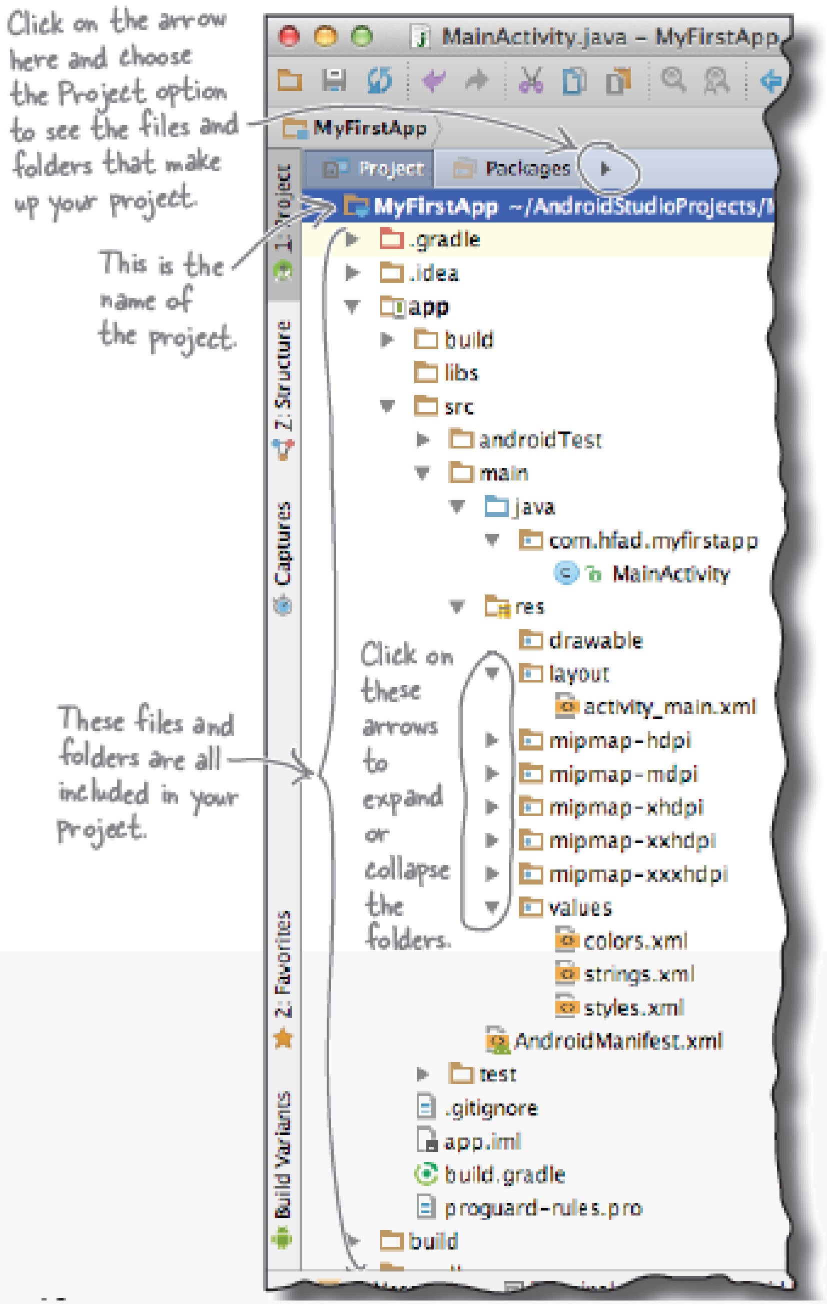
- The Android Studio wizard created a project for your app, configured to your specifications.
- It created a basic activity and layout with template code.



```
package com.hfad.myfirstapp;

import ...

public class MainActivity extends Activity {

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

Gradle build finished in 21s 987ms (4 minutes ago)    1:1  LF  UTF-8  Context: <no context>

# A complete Folder Structure


Click on the arrow here and choose the Project option to see the files and folders that make up your project.

This is the name of the project.

These files and folders are all included in your project.

Click on these arrows to expand or collapse the folders.

## The Folder Structure

- **Java and XML Files**

  These are the activity and layout files the wizard created for you.
- **Android-generated Java Files**

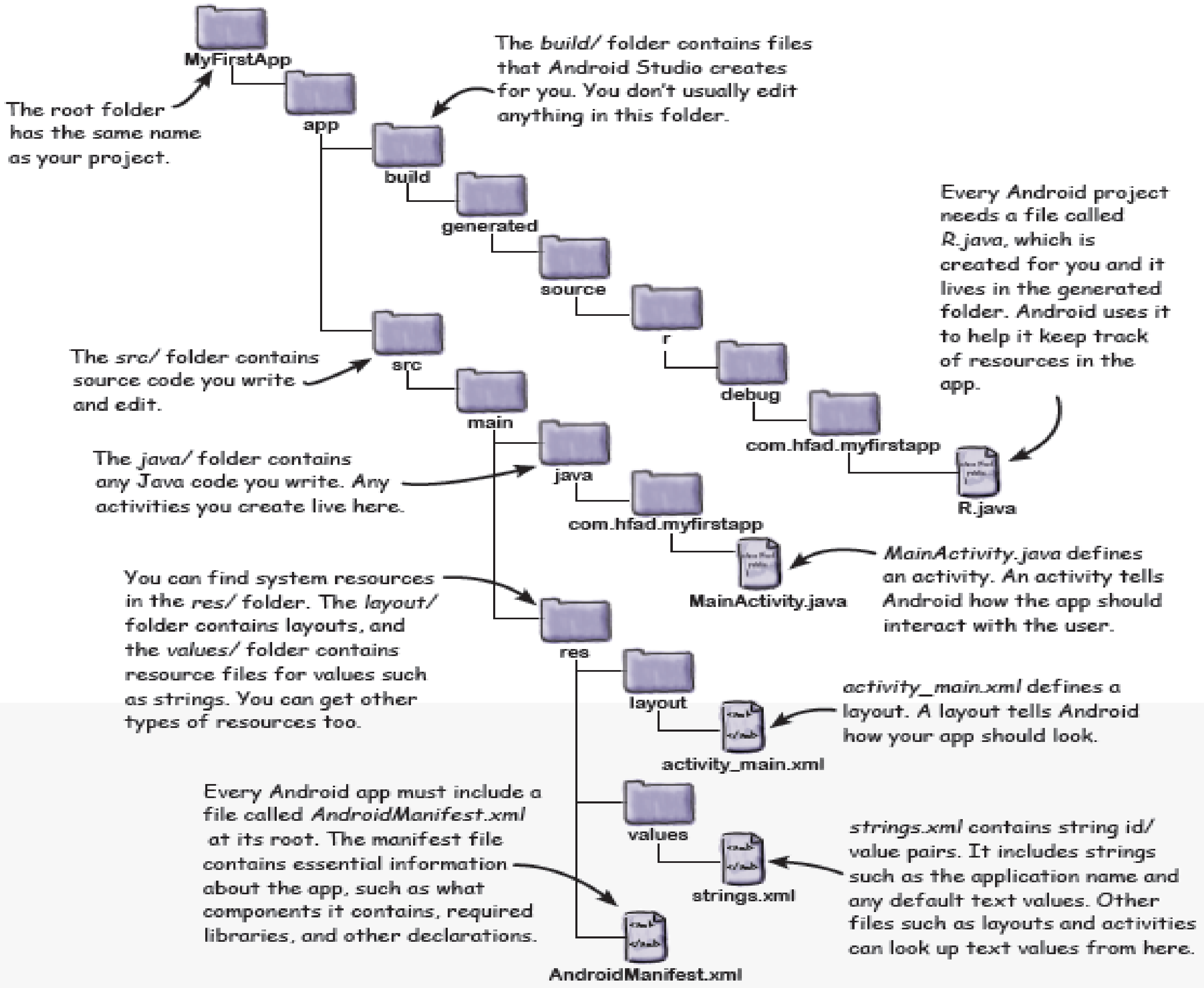
  There are some extra Java files you don’t need to touch which Android Studio generates for you automatically.
- **Resources Files**

  These include default image files for icons, styles your app might use, and any common String values your app might want to look up.
- **Android Libraries**

  In the wizard, you specified the minimum SDK version you want your app to be compatible with. Android Studio makes sure it includes the relevant Android libraries for this version.
- **Configuration Files**

  The configuration files tell Android what’s actually in the app and how it should run.

# Useful Files in Your Project

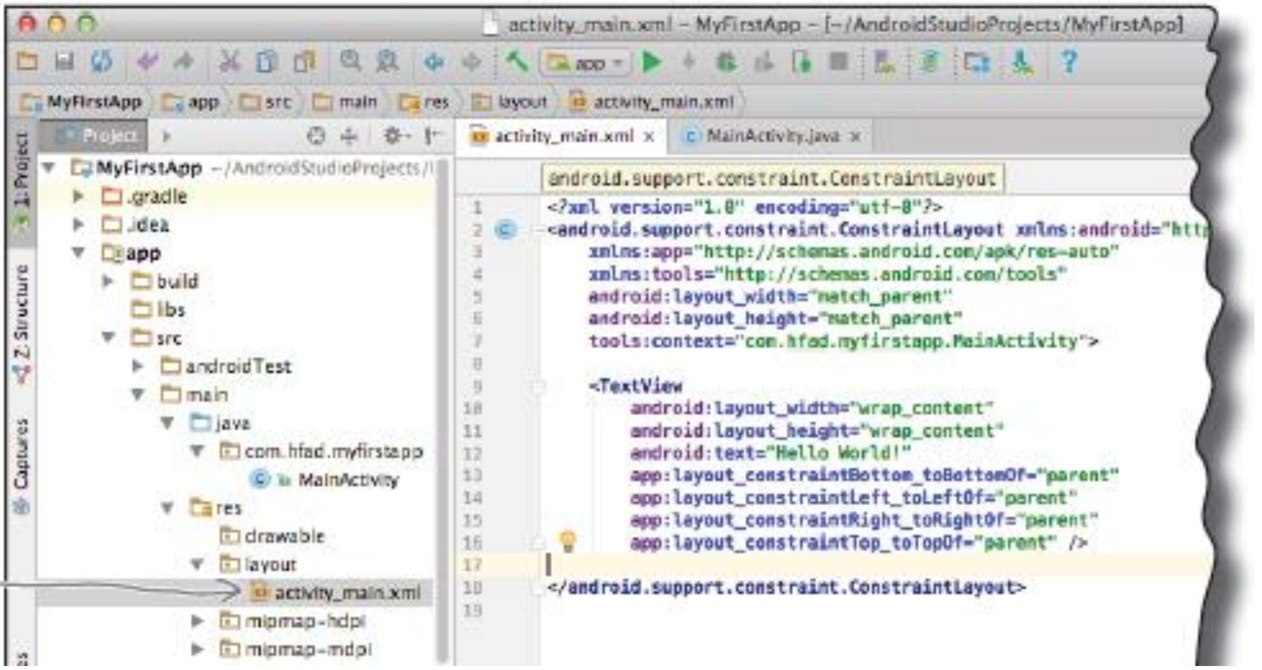

MyFirstApp
app
build
generated
source
r
debug
com.hfad.myfirstapp
R.java
src
main
java
com.hfad.myfirstapp
MainActivity.java
res
layout
activity_main.xml
values
strings.xml
AndroidManifest.xml
The root folder has the same name as your project.
The *build/* folder contains files that Android Studio creates for you. You don't usually edit anything in this folder.
Every Android project needs a file called *R.java*, which is created for you and it lives in the generated folder. Android uses it to help it keep track of resources in the app.
The *src/* folder contains source code you write and edit.
The *java/* folder contains any Java code you write. Any activities you create live here.
*MainActivity.java* defines an activity. An activity tells Android how the app should interact with the user.
You can find system resources in the *res/* folder. The *layout/* folder contains layouts, and the *values/* folder contains resource files for values such as strings. You can get other types of resources too.
*activity_main.xml* defines a layout. A layout tells Android how your app should look.
Every Android app must include a file called *AndroidManifest.xml* at its root. The manifest file contains essential information about the app, such as what components it contains, required libraries, and other declarations.
*strings.xml* contains string id/value pairs. It includes strings such as the application name and any default text values. Other files such as layouts and activities can look up text values from here.

# Edit Code

## The code editor

Most files get displayed in the code editor, which is just like a text editor, but with extra features such as color coding and code checking.

Double-click on the file in the explorer and the file contents appear in the editor panel.

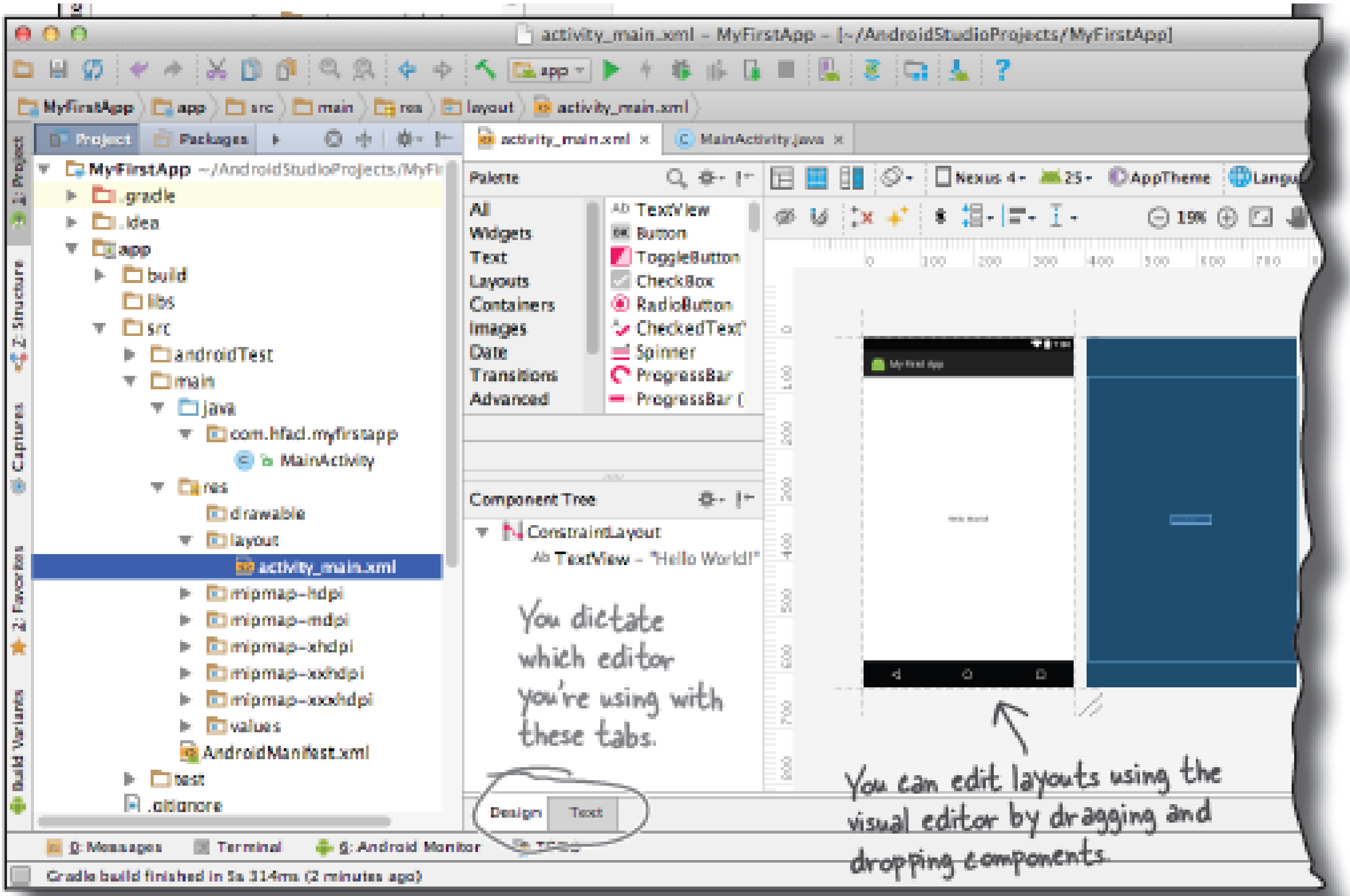


```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.hfad.myfirstapp.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

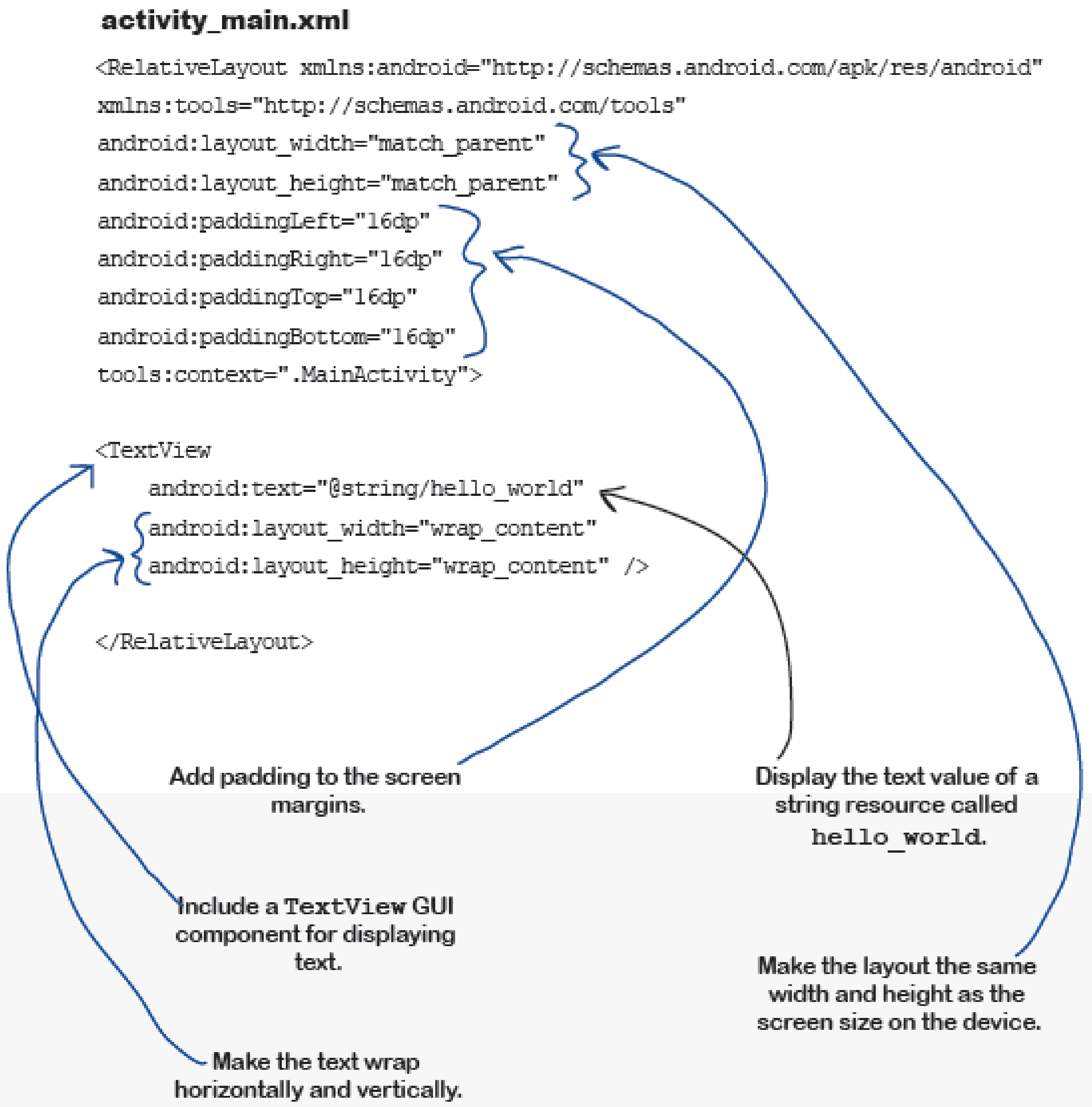
# Edit Code

## The design editor

If you're editing a layout, you have an extra option. Rather than edit the XML (such as that shown on the next page), you can use the design editor, which allows you to drag GUI components onto your layout, and arrange them how you want. The code editor and design editor give different views of the same file, so you can switch back and forth between the two.



# Activity_main.xml



**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="16dp"
android:paddingRight="16dp"
android:paddingTop="16dp"
android:paddingBottom="16dp"
tools:context=".MainActivity">

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

</RelativeLayout>
```

Add padding to the screen margins.

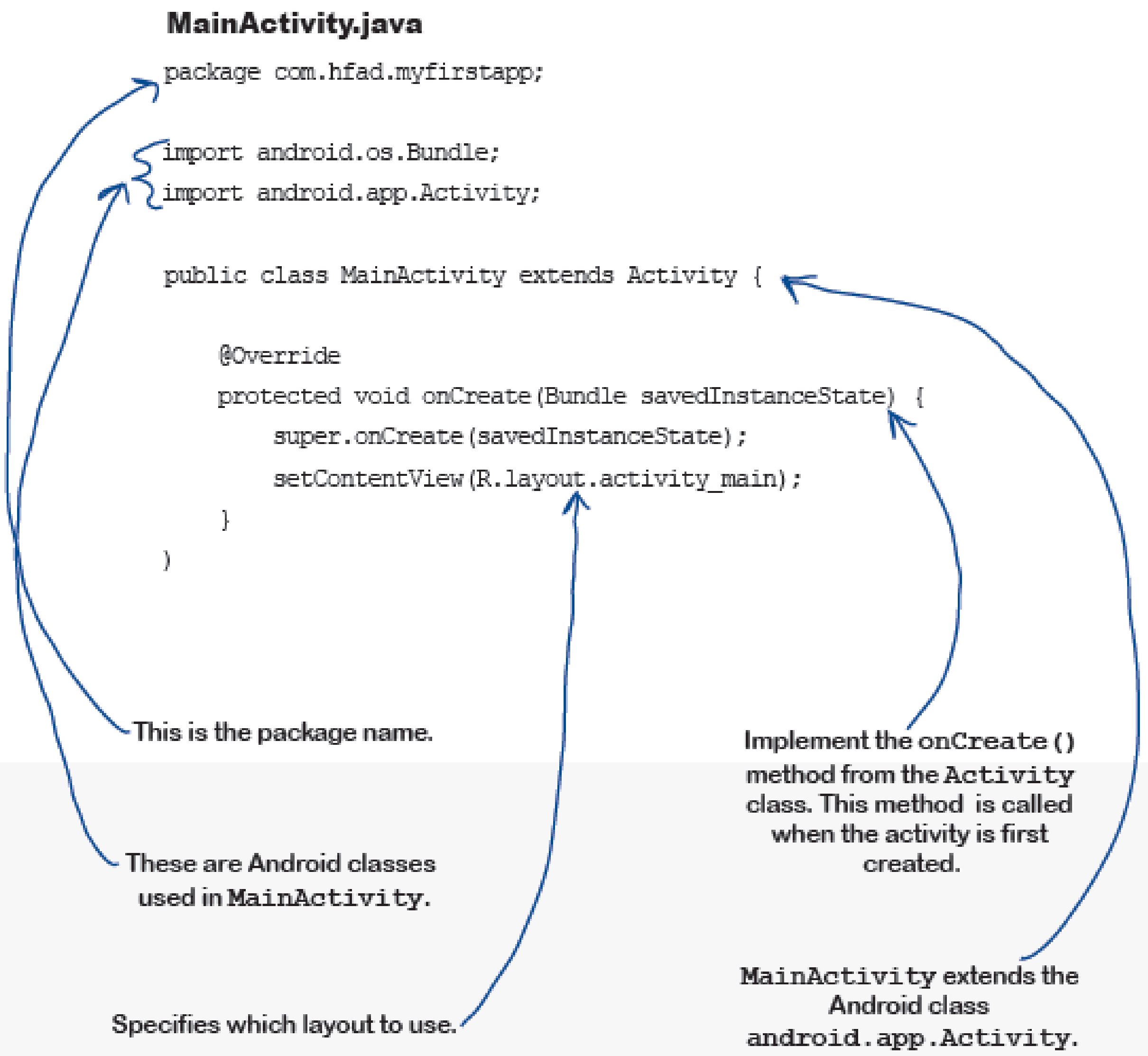
Display the text value of a string resource called `hello_world`.

Include a `TextView` GUI component for displaying text.

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

Make the text wrap horizontally and vertically.

# MainActivity.java



**MainActivity.java**

```
package com.hfad.myfirstapp;

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

public class MainActivity extends Activity {

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

This is the package name.

These are Android classes used in `MainActivity`.

Specifies which layout to use.

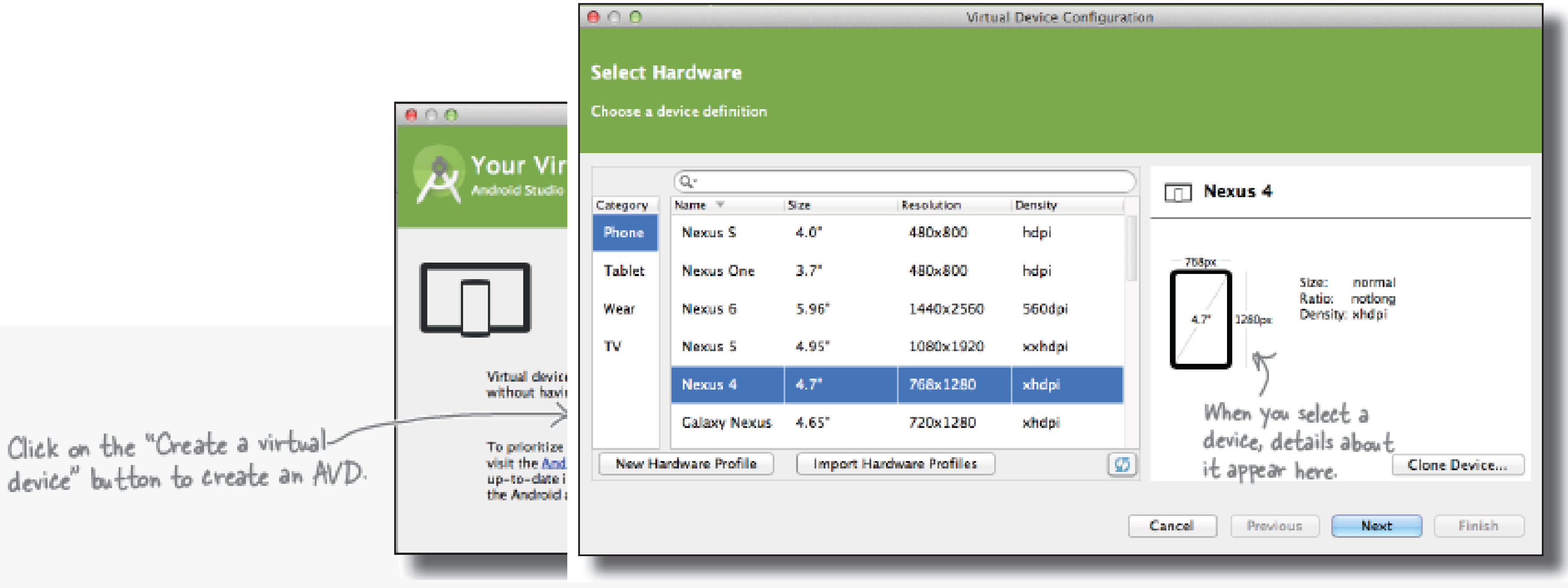
Implement the `onCreate()` method from the `Activity` class. This method is called when the activity is first created.

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

# Getting Started – Run App

## 1. Open AVD and Select the Hardware

- The Android emulator allows you to run your app on an Android Virtual Device (AVD).
- AVD behaves just like physical Android device.
- You can set up numerous AVDs,each emulating a different type of devices.



Click on the "Create a virtual device" button to create an AVD.

Virtual Device Configuration

Select Hardware

Choose a device definition

| Category | Name | Size | Resolution | Density |
|---|---|---|---|---|
| Phone | Nexus S | 4.0" | 480x800 | hdpi |
| Tablet | Nexus One | 3.7" | 480x800 | hdpi |
| Wear | Nexus 6 | 5.96" | 1440x2560 | 560dpi |
| TV | Nexus 5 | 4.95" | 1080x1920 | xxhdpi |
| | Nexus 4 | 4.7" | 768x1280 | xhdpi |
| | Galaxy Nexus | 4.65" | 720x1280 | xhdpi |

Nexus 4

768px, 4.7", 1280px

Size: normal
Ratio: notlong
Density: xhdpi

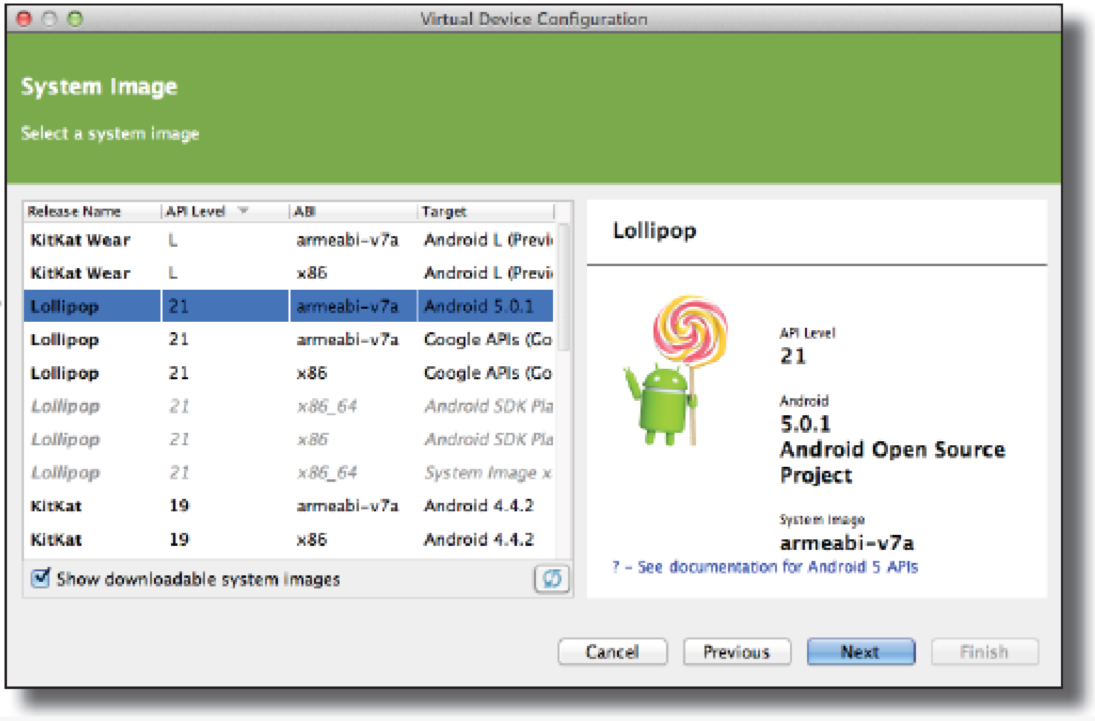
When you select a device, details about it appear here.

# Getting Started – Run App

## 2. Creating an AVD

### Select a system Image

If you don't have this system image installed, you'll be given the option to download it.



Virtual Device Configuration

System Image

Select a system image

| Release Name | API Level | ABI | Target |
|---|---|---|---|
| KitKat Wear | L | armeabi-v7a | Android L (Previ |
| KitKat Wear | L | x86 | Android L (Previ |
| Lollipop | 21 | armeabi-v7a | Android 5.0.1 |
| Lollipop | 21 | armeabi-v7a | Google APIs (Go |
| Lollipop | 21 | x86 | Google APIs (Go |
| *Lollipop* | *21* | *x86_64* | *Android SDK Pla* |
| *Lollipop* | *21* | *x86* | *Android SDK Pla* |
| *Lollipop* | *21* | *x86_64* | *System Image x* |
| KitKat | 19 | armeabi-v7a | Android 4.4.2 |
| KitKat | 19 | x86 | Android 4.4.2 |

[✓] Show downloadable system images

Lollipop

API Level
21

Android
5.0.1
Android Open Source Project

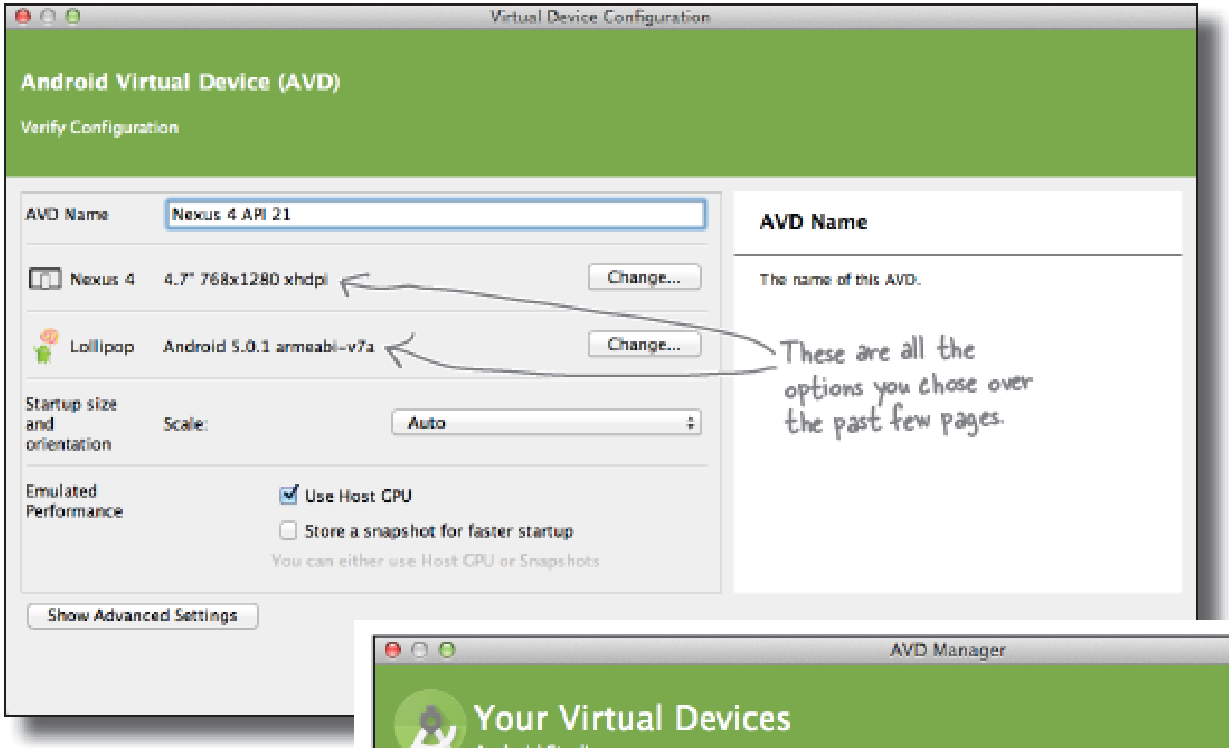
System Image
armeabi-v7a

? - See documentation for Android 5 APIs

# Getting Started – Run App

## 2. Creating an AVD

### Verify the AVD configuration



Virtual Device Configuration

**Android Virtual Device (AVD)**

Verify Configuration

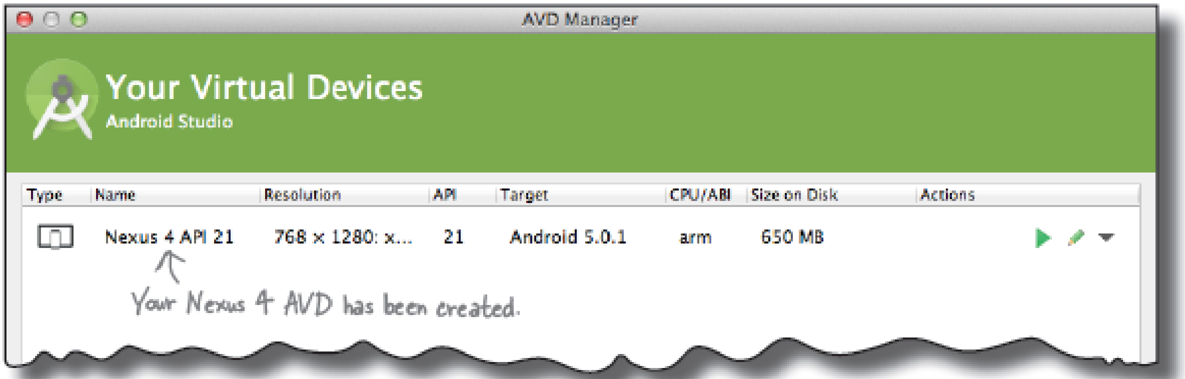
| | | |
|---|---|---|
| AVD Name | Nexus 4 API 21 | |
| Nexus 4 | 4.7" 768x1280 xhdpi | Change... |
| Lollipop | Android 5.0.1 armeabi-v7a | Change... |
| Startup size and orientation | Scale: Auto | |
| Emulated Performance | ☑ Use Host GPU<br>☐ Store a snapshot for faster startup<br>You can either use Host GPU or Snapshots | |

Show Advanced Settings

**AVD Name**

The name of this AVD.

These are all the options you chose over the past few pages.



AVD Manager

Your Virtual Devices

Android Studio

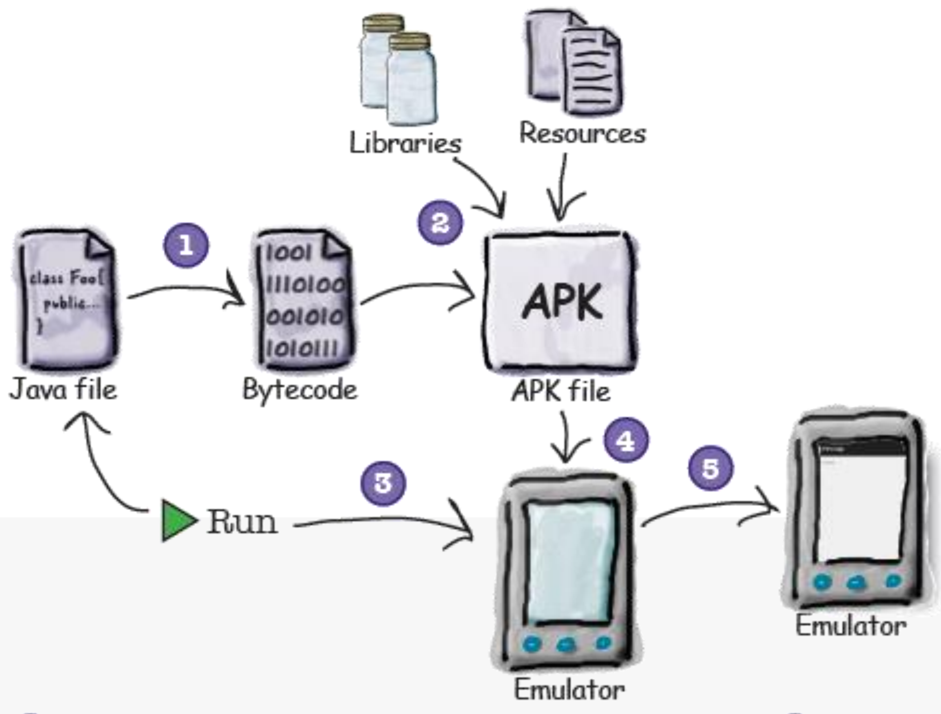
| Type | Name | Resolution | API | Target | CPU/ABI | Size on Disk | Actions |
|---|---|---|---|---|---|---|---|
| | Nexus 4 API 21 | 768 × 1280: x... | 21 | Android 5.0.1 | arm | 650 MB | |

Your Nexus 4 AVD has been created.

# Getting Started – Run App

## 3. Compile, Package, Deploy, and Run

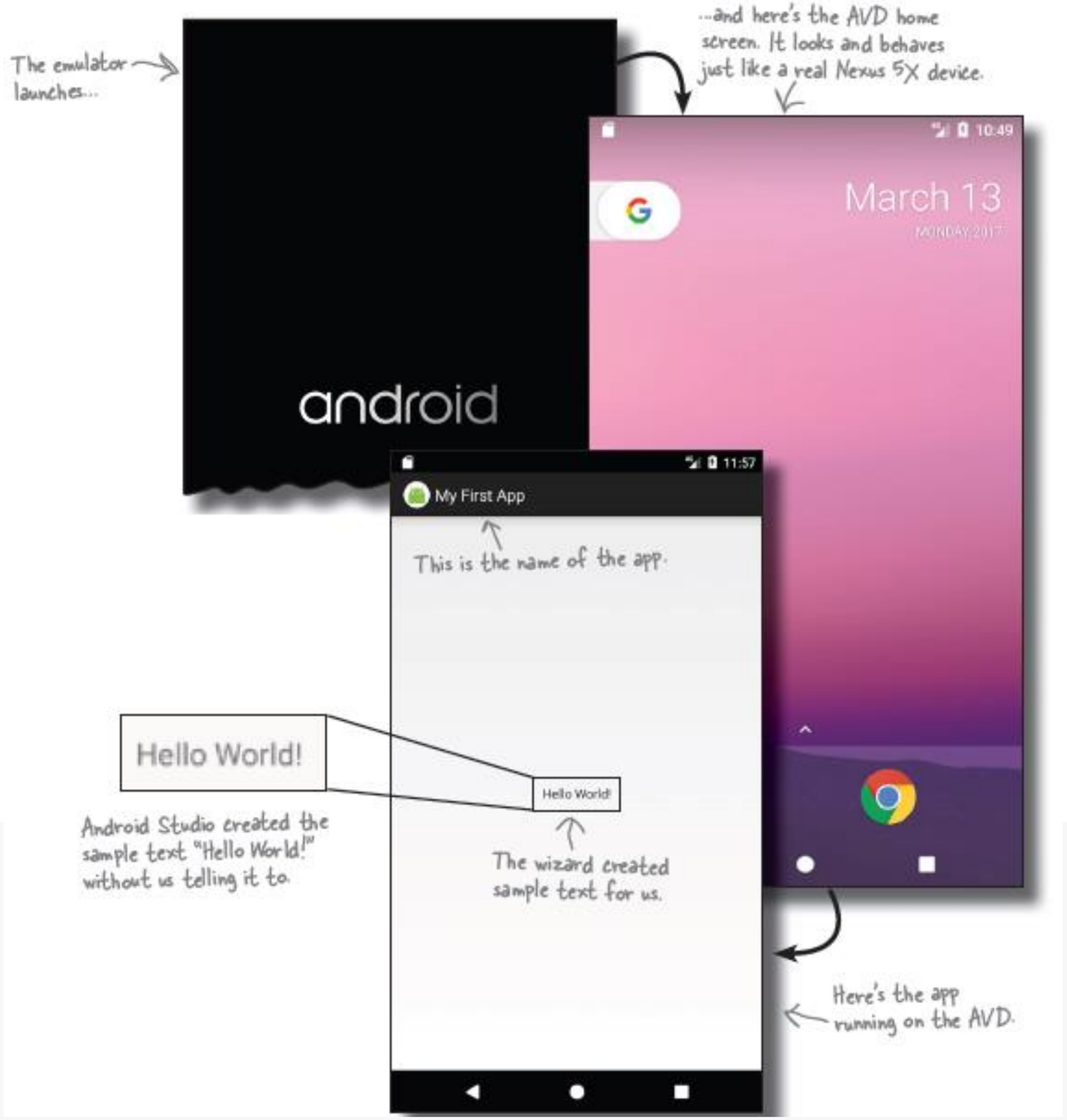
- An APK File is an application package. It's basically a JAR or ZIP file for android application.



1. **The Java source files get compiled to bytecode.**
2. **An Android application package, or APK file, gets created.** The APK file includes the compiled Java files, along with any libraries and resources needed by your app.
3. **Assuming there's not one already running, the emulator gets launched with the AVD.**
4. **Once the emulator has been launched and the AVD is active, the APK file is uploaded to the AVD and installed.**
5. **The AVD starts the main activity associated with the app.** Your app gets displayed on the AVD screen, and it's all ready for you to test out.

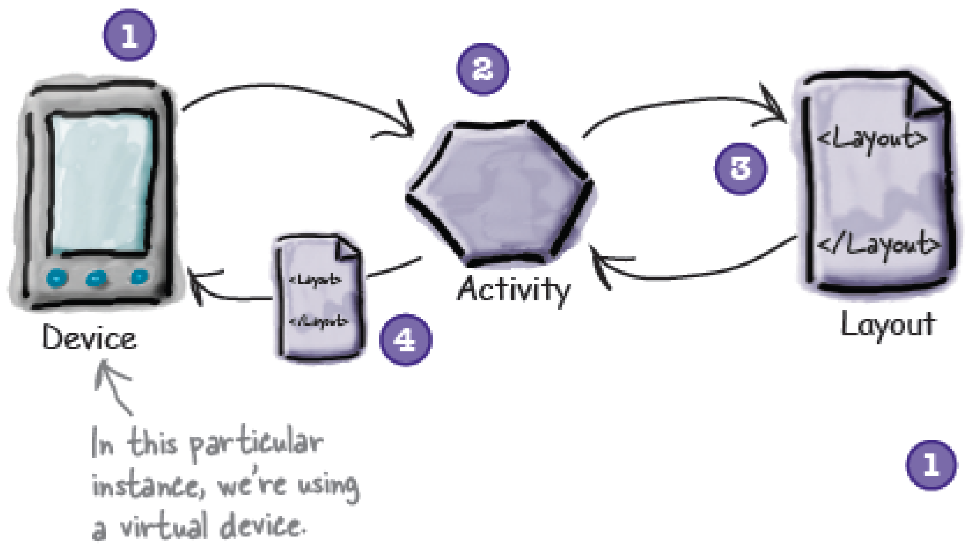
## Getting Started – Run App

### 4. Test Drive


The emulator launches...
android
...and here's the AVD home screen. It looks and behaves just like a real Nexus 5X device.
10:49
March 13
MONDAY 2017
11:57
My First App
This is the name of the app.
Hello World!
Hello World!
Android Studio created the sample text "Hello World!" without us telling it to.
The wizard created sample text for us.
Here's the app running on the AVD.

# Getting Started – Run App

## 4. Test Drive (Cont.)



1. Android Studio launches the emulator, loads the AVD, and installs the app.

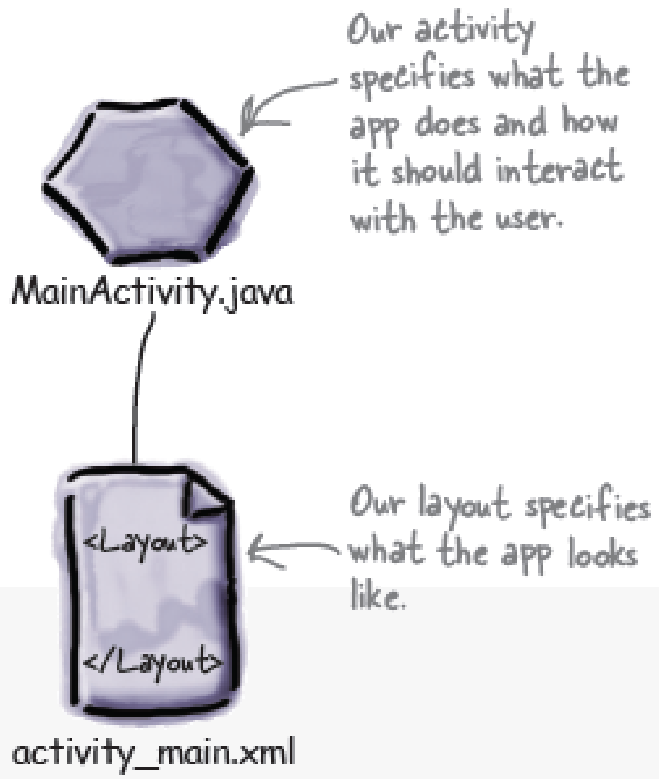
2. When the app gets launched, an activity object is created from MainActivity.java.

3. The activity specifies that it uses the layout activity_main.xml.

4. The activity tells Android to display the layout on the screen. The text "Hello world!" gets displayed.

## Getting Started – Refining App

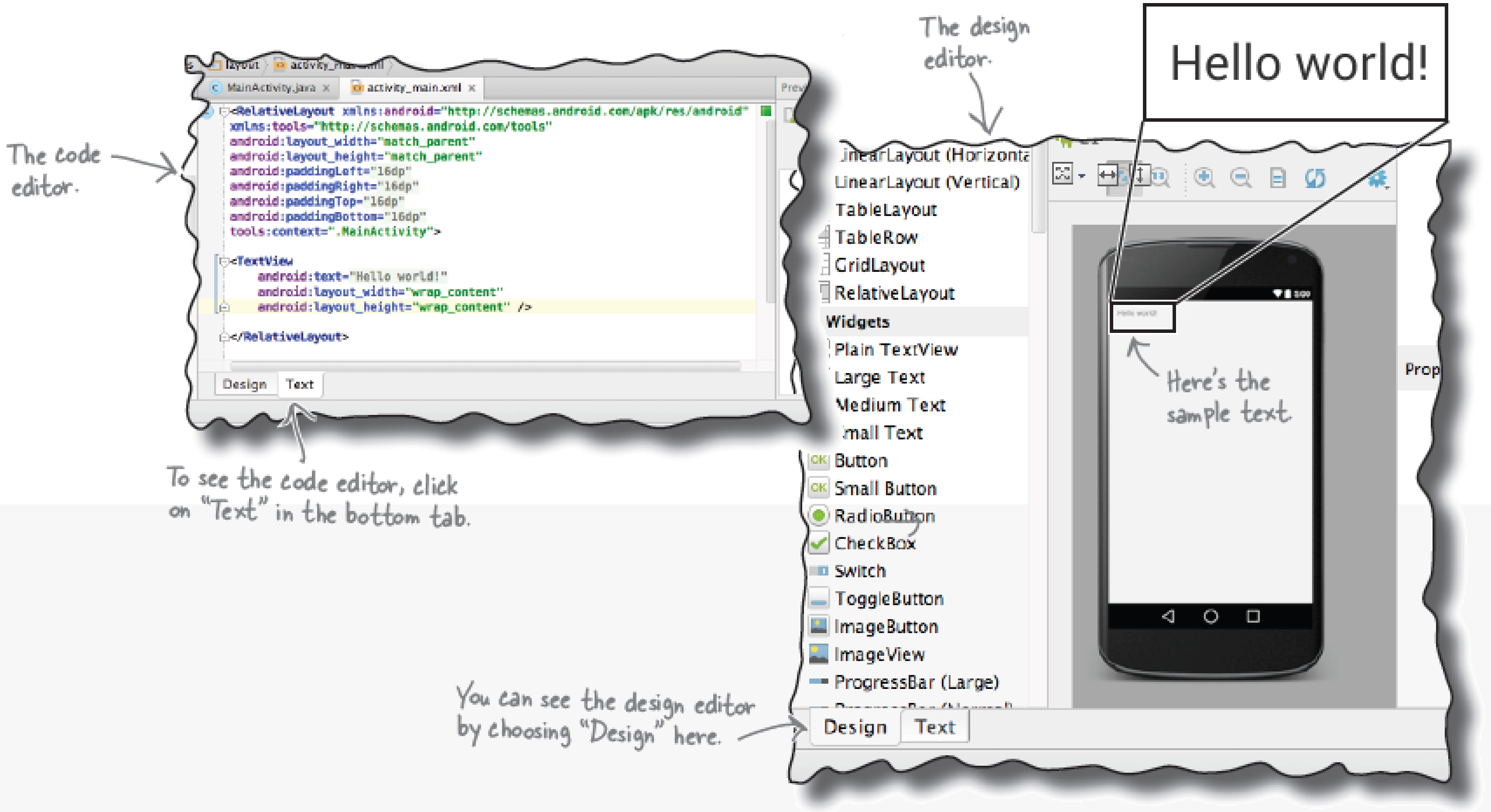
- The app has one activity and one layout


Our activity specifies what the app does and how it should interact with the user.
MainActivity.java
<Layout>
</Layout>
Our layout specifies what the app looks like.
activity_main.xml

# Getting Started – Refining App

## What's in the Layout?

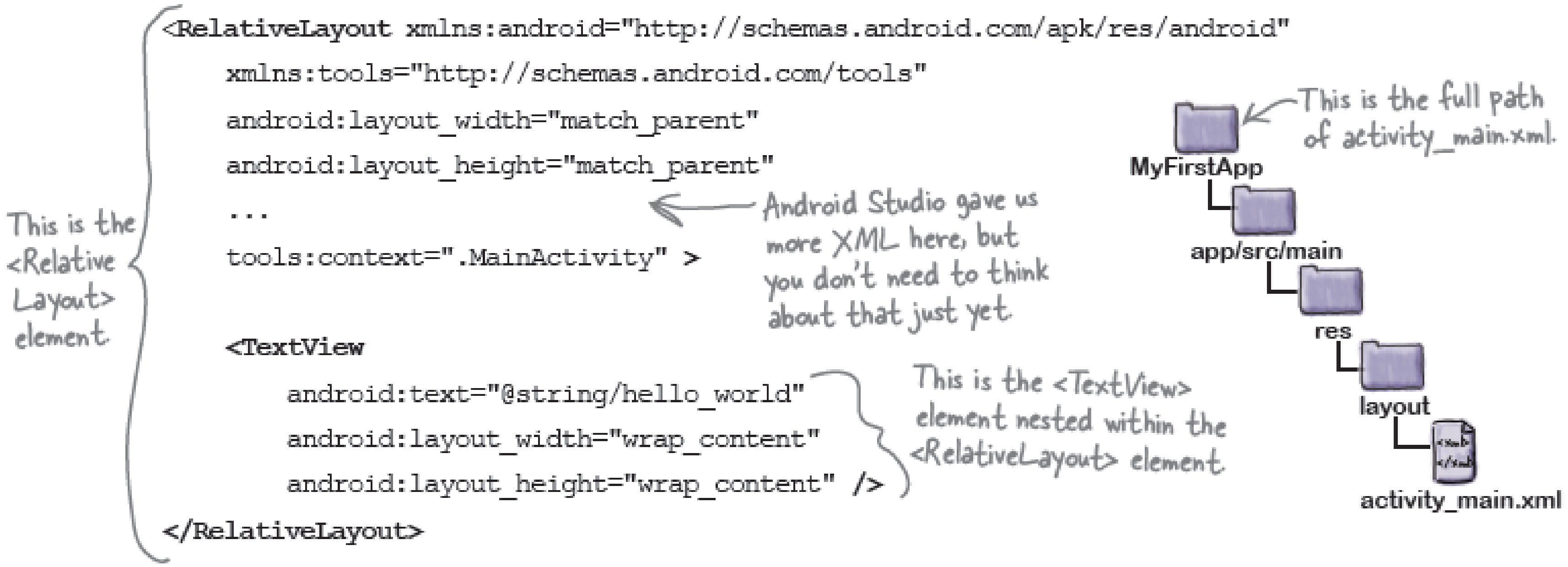
- There are two ways of viewing and editing layout files in Android studio through the **code editor** and **design editor**



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

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

</RelativeLayout>
```

# Getting Started – Refining App

## Activity_main.xml has 2 Elements



```
<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"
    ...
    tools:context=".MainActivity" >

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

This is the <RelativeLayout> element

Android Studio gave us more XML here, but you don't need to think about that just yet.

This is the <TextView> element nested within the <RelativeLayout> element.

This is the full path of activity_main.xml.

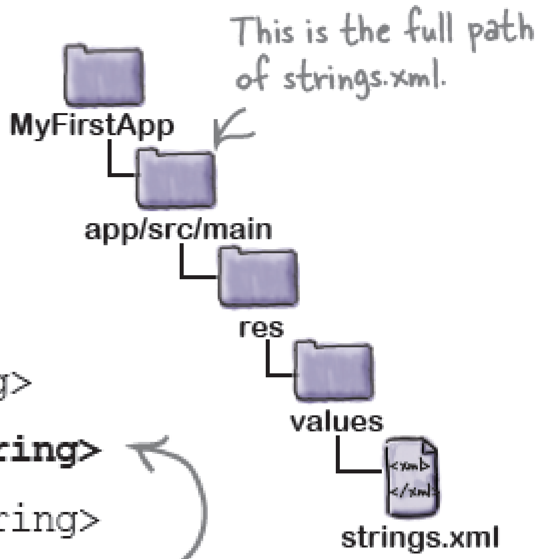
## Getting Started – Refining App

### Strings.xml

Display the text...

...for string resource hello_world.

```
android:text="@string/hello_world" />
```

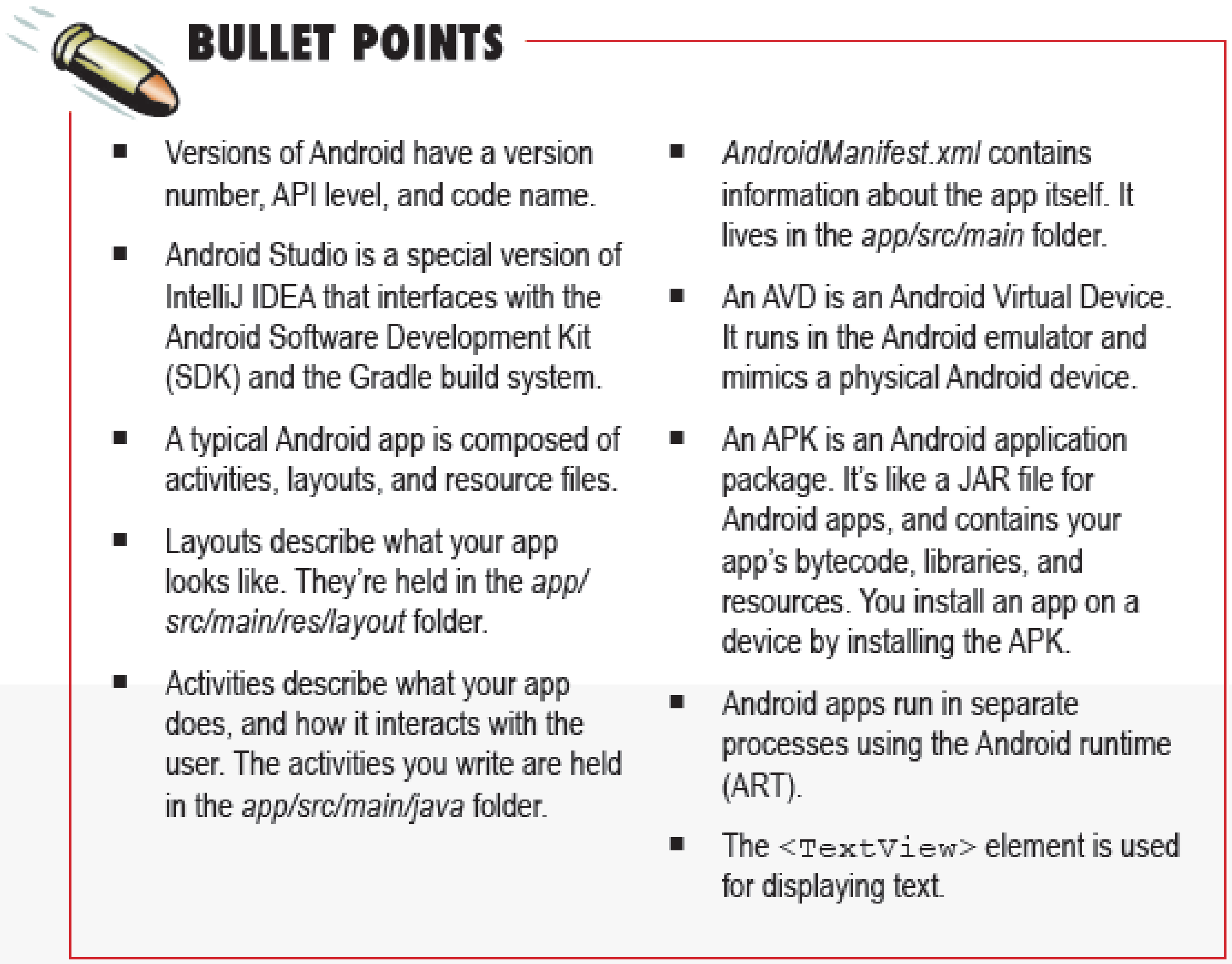


```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">My First App</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
</resources>
```

strings.xml includes a string with a name of hello_world, and a value of "Hello world!".

- Put string values in string.xml rather than hardcoding them.
- String.xml is a resource file used to hold name/ value pairs of strings.

# Summary



**BULLET POINTS**

- Versions of Android have a version number, API level, and code name.
- Android Studio is a special version of IntelliJ IDEA that interfaces with the Android Software Development Kit (SDK) and the Gradle build system.
- A typical Android app is composed of activities, layouts, and resource files.
- Layouts describe what your app looks like. They're held in the *app/src/main/res/layout* folder.
- Activities describe what your app does, and how it interacts with the user. The activities you write are held in the *app/src/main/java* folder.
- *AndroidManifest.xml* contains information about the app itself. It lives in the *app/src/main* folder.
- An AVD is an Android Virtual Device. It runs in the Android emulator and mimics a physical Android device.
- An APK is an Android application package. It's like a JAR file for Android apps, and contains your app's bytecode, libraries, and resources. You install an app on a device by installing the APK.
- Android apps run in separate processes using the Android runtime (ART).
- The `<TextView>` element is used for displaying text.

## References

- Head First Android Development. 2nd Edition. A Brain friendy guide. Dawn Griffiths and David Griffiths.O'reilly. ISBN:978-1-491-97405-6. Chapter 1
- https://developer.android.com/