IT332: Mobile Application Development

Lecture # 02: Getting Started

Muhammad Imran

Outline

- The Android Developer Community
- Android SDK
- Android SDK Tools using Command-line
- Android Virtual Devices
- Testing on Physical Android Device
- Creating a New Project
- Launching your First Android App
- App Manifest Overview

The Android Developer Community

- The following are some developer communities and websites that you can turn to for help if you run into problems while working with Android:

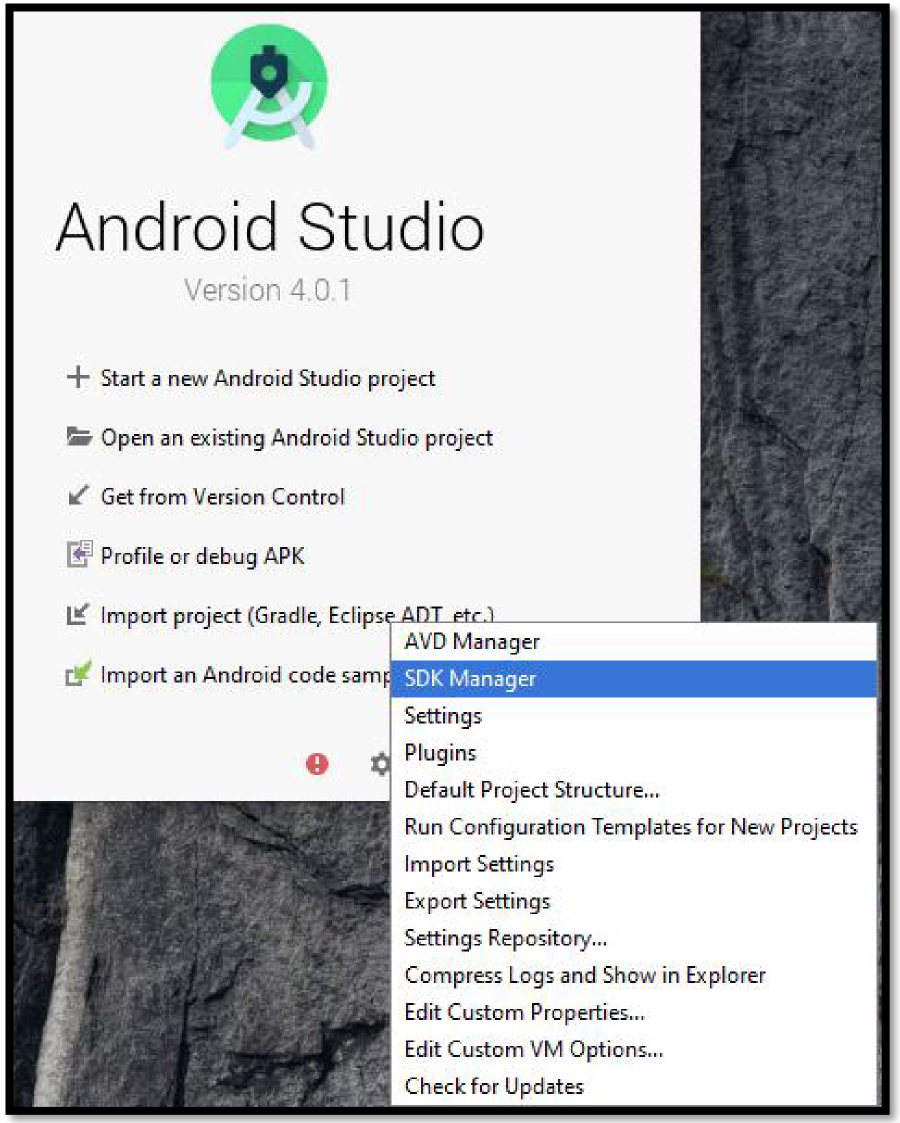
- Stack Overflow (www.stackoverflow.com)— Stack Overflow is a collaboratively edited question-and-answer site for developers.

- Google Android Training (http://developer.android.com/training/index.html)— Google has launched the Android Training site, which contains a number of useful classes grouped by topics.

- Android Discuss (http://groups.google.com/group/android-discuss)— Android Discuss is a discussion group hosted by Google using the Google Groups service. Here, you will be able to discuss the various aspects of Android programming.

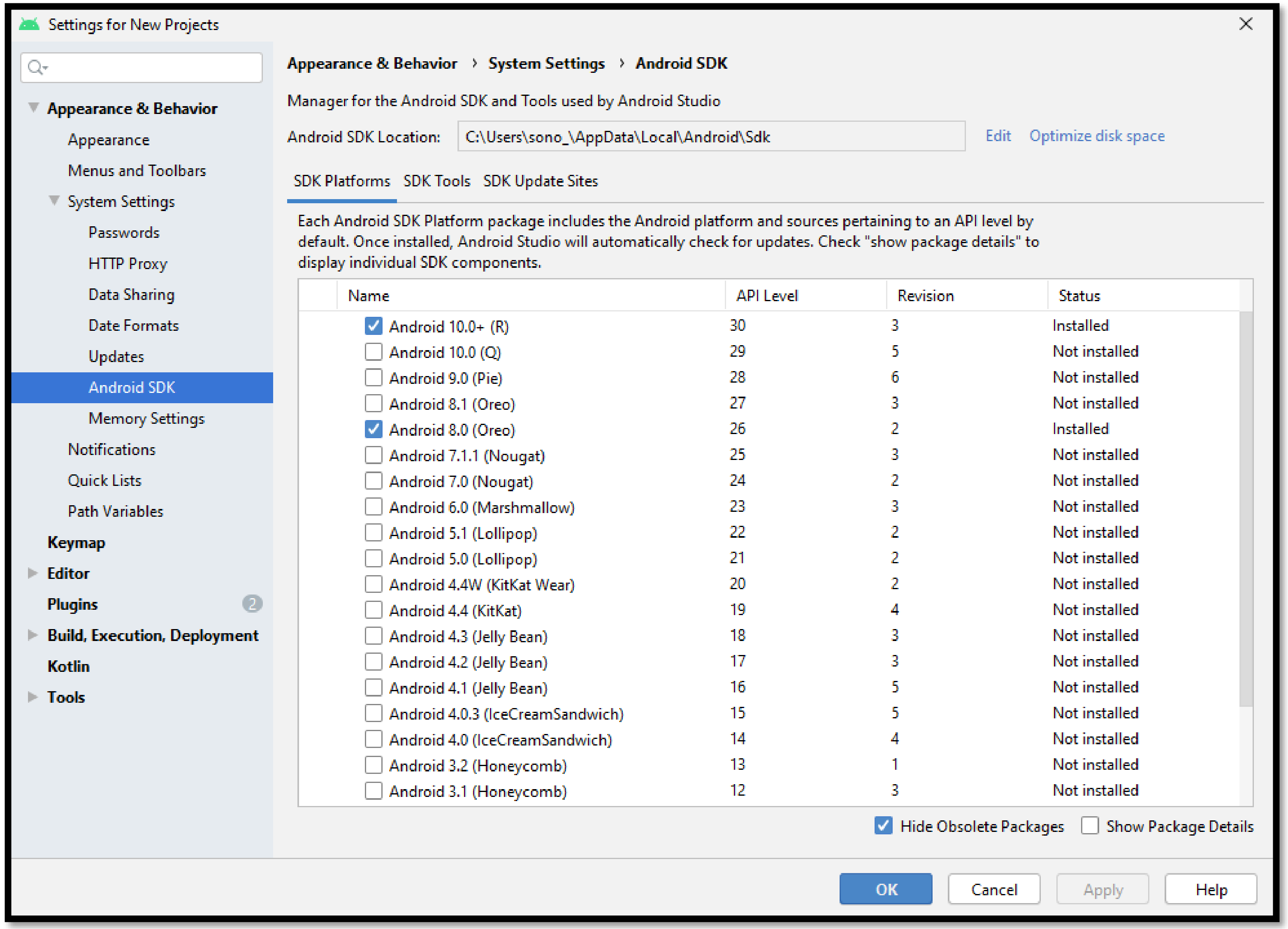
Android SDK

- The most important piece of software you need to download is, of course, the Android SDK.
- The SDKs are named after the version of Android OS to which they correspond.
- By default, the Android 10.0+ SDK was installed with Android Studio 4.0.
- By using the SDK Manager we can install a different Android SDK



Android Studio
Version 4.0.1
Start a new Android Studio project
Open an existing Android Studio project
Get from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code samp
AVD Manager
SDK Manager
Settings
Plugins
Default Project Structure...
Run Configuration Templates for New Projects
Import Settings
Export Settings
Settings Repository...
Compress Logs and Show in Explorer
Edit Custom Properties...
Edit Custom VM Options...
Check for Updates

Android SDK Manager



Settings for New Projects

- Appearance & Behavior
  - Appearance
  - Menus and Toolbars
  - System Settings
    - Passwords
    - HTTP Proxy
    - Data Sharing
    - Date Formats
    - Updates
    - **Android SDK**
    - Memory Settings
  - Notifications
  - Quick Lists
  - Path Variables
- Keymap
- Editor
- Plugins 2
- Build, Execution, Deployment
- Kotlin
- Tools

Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: C:\Users\sono_\AppData\Local\Android\Sdk   Edit   Optimize disk space

SDK Platforms | SDK Tools | SDK Update Sites

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.

	Name	API Level	Revision	Status
✓	Android 10.0+ (R)	30	3	Installed
☐	Android 10.0 (Q)	29	5	Not installed
☐	Android 9.0 (Pie)	28	6	Not installed
☐	Android 8.1 (Oreo)	27	3	Not installed
✓	Android 8.0 (Oreo)	26	2	Installed
☐	Android 7.1.1 (Nougat)	25	3	Not installed
☐	Android 7.0 (Nougat)	24	2	Not installed
☐	Android 6.0 (Marshmallow)	23	3	Not installed
☐	Android 5.1 (Lollipop)	22	2	Not installed
☐	Android 5.0 (Lollipop)	21	2	Not installed
☐	Android 4.4W (KitKat Wear)	20	2	Not installed
☐	Android 4.4 (KitKat)	19	4	Not installed
☐	Android 4.3 (Jelly Bean)	18	3	Not installed
☐	Android 4.2 (Jelly Bean)	17	3	Not installed
☐	Android 4.1 (Jelly Bean)	16	5	Not installed
☐	Android 4.0.3 (IceCreamSandwich)	15	5	Not installed
☐	Android 4.0 (IceCreamSandwich)	14	4	Not installed
☐	Android 3.2 (Honeycomb)	13	1	Not installed
☐	Android 3.1 (Honeycomb)	12	3	Not installed

✓ Hide Obsolete Packages   ☐ Show Package Details

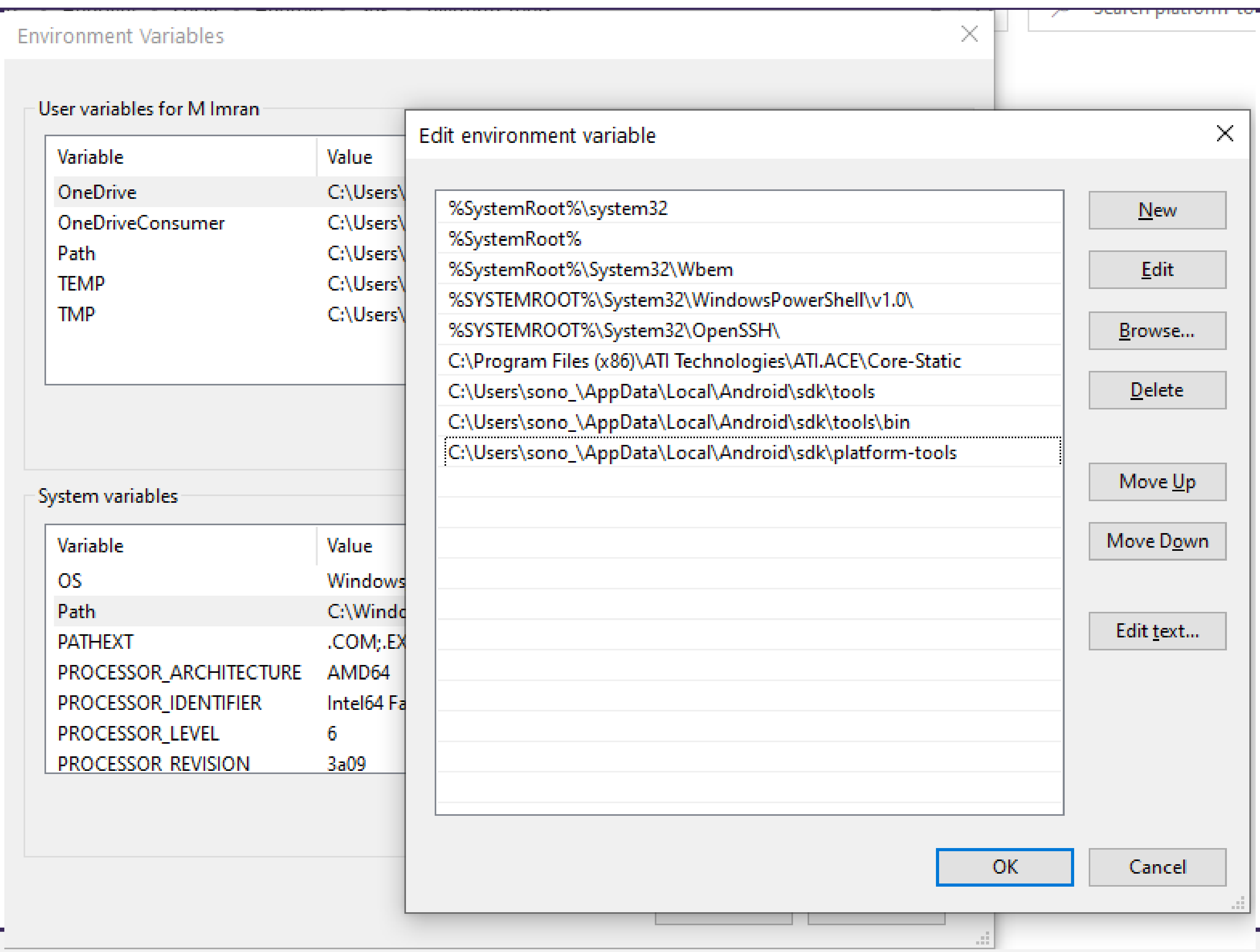
OK | Cancel | Apply | Help

Android SDK Tools using Command-line

- The underlying tools of the Android SDK can be accessed from within the Android Studio
- It can also be useful to invoke those tools from a command prompt.
- For the operating system to be able to find these tools, we add them to the system's PATH environment variable.
- The PATH variable needs to be configured to include the following paths (where <path_to_android_sdk_installation> represents the file system location into which the Android SDK was installed):

```
<path_to_android_sdk_installation>/sdk/tools
<path_to_android_sdk_installation>/sdk/tools/bin
<path_to_android_sdk_installation>/sdk/platform-tools
```

Setting the Path Variable



Environment Variables

User variables for M Imran

Variable	Value
OneDrive	C:\Users\
OneDriveConsumer	C:\Users\
Path	C:\Users\
TEMP	C:\Users\
TMP	C:\Users\

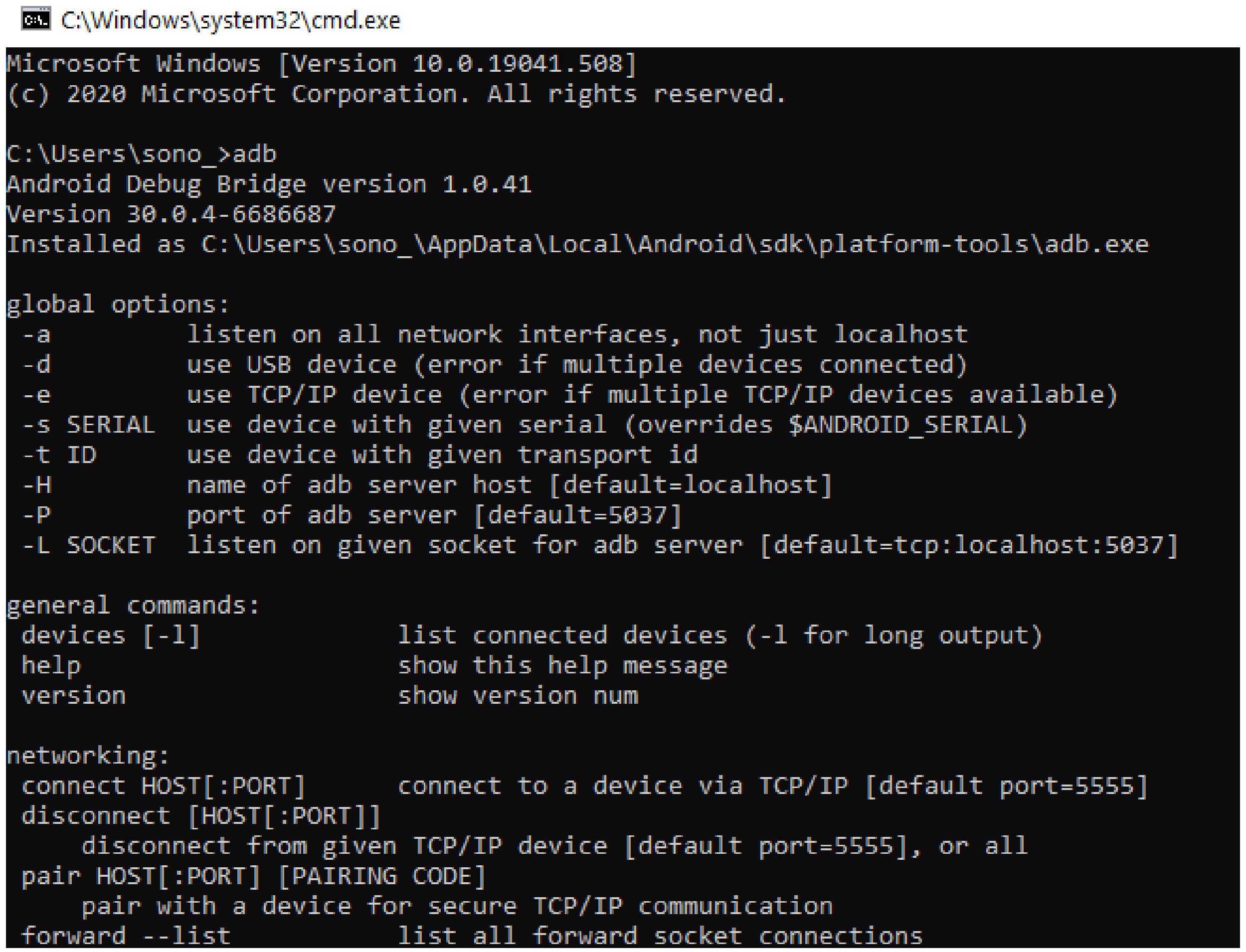
System variables

Variable	Value
OS	Windows
Path	C:\Windo
PATHEXT	.COM;.EX
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Fa
PROCESSOR_LEVEL	6
PROCESSOR_REVISION	3a09

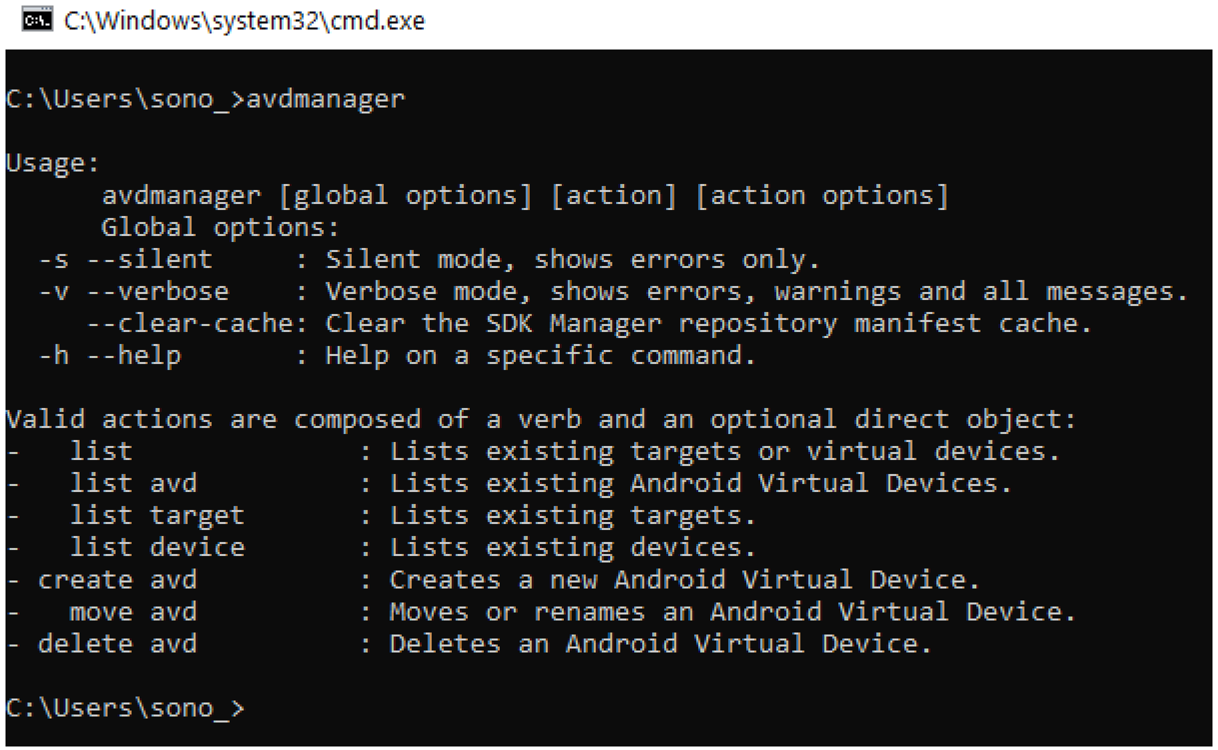
Edit environment variable

```
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files (x86)\ATI Technologies\ATI.ACE\Core-Static
C:\Users\sono_\AppData\Local\Android\sdk\tools
C:\Users\sono_\AppData\Local\Android\sdk\tools\bin
C:\Users\sono_\AppData\Local\Android\sdk\platform-tools
```

Android SDK Tools using Command-line



```
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\sono_>adb
Android Debug Bridge version 1.0.41
Version 30.0.4-6686687
Installed as C:\Users\sono_\AppData\Local\Android\sdk\platform-tools\adb.exe

global options:
 -a         listen on all network interfaces, not just localhost
 -d         use USB device (error if multiple devices connected)
 -e         use TCP/IP device (error if multiple TCP/IP devices available)
 -s SERIAL  use device with given serial (overrides $ANDROID_SERIAL)
 -t ID      use device with given transport id
 -H         name of adb server host [default=localhost]
 -P         port of adb server [default=5037]
 -L SOCKET  listen on given socket for adb server [default=tcp:localhost:5037]

general commands:
 devices [-l]             list connected devices (-l for long output)
 help                     show this help message
 version                  show version num

networking:
 connect HOST[:PORT]      connect to a device via TCP/IP [default port=5555]
 disconnect [HOST[:PORT]]
     disconnect from given TCP/IP device [default port=5555], or all
 pair HOST[:PORT] [PAIRING CODE]
     pair with a device for secure TCP/IP communication
 forward --list           list all forward socket connections
```



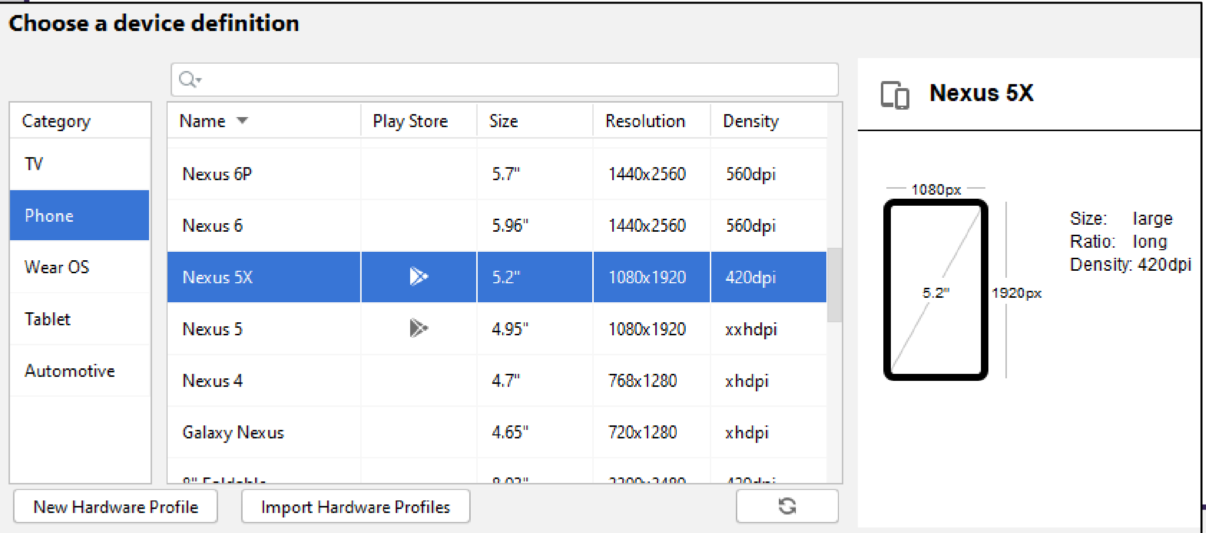
```
C:\Users\sono_>avdmanager

Usage:
      avdmanager [global options] [action] [action options]
      Global options:
  -s --silent     : Silent mode, shows errors only.
  -v --verbose    : Verbose mode, shows errors, warnings and all messages.
     --clear-cache: Clear the SDK Manager repository manifest cache.
  -h --help       : Help on a specific command.

Valid actions are composed of a verb and an optional direct object:
-   list              : Lists existing targets or virtual devices.
-   list avd          : Lists existing Android Virtual Devices.
-   list target       : Lists existing targets.
-   list device       : Lists existing devices.
- create avd          : Creates a new Android Virtual Device.
-   move avd          : Moves or renames an Android Virtual Device.
- delete avd          : Deletes an Android Virtual Device.

C:\Users\sono_>
```

Creating Android Virtual Devices (AVDs)

- An AVD is an emulator instance that enables you to model an actual device for testing of Android apps
- Each AVD consists of a hardware profile; a mapping to a system image; and emulated storage, such as a secure digital (SD) card.
- Emulators are not perfect but good for doing some generalized testing of your applications.
- Applications, such as games (which are GPU heavy) or applications that use sensors such as the GPS or accelerometer cannot be simulated with the same speed or consistency within an emulator
- You can create as many AVDs as you want to test your applications with different configurations.
- This testing is important to confirm the behavior of your application when it is run on different devices with varying capabilities.

Creating Android Virtual Devices (AVDs)

- Launch the AVD Manager from tools menu or by using the AVD Manager button
- You can use the Nexus 5x hardware profile for AVD
- None of the emulators offers the same performance as its actual hardware counterpart
- Nexus 5x should run well on most x86-based desktops, and it still offers some of the mid- to high-end Android device specs.
- For the system image, select and install the latest option



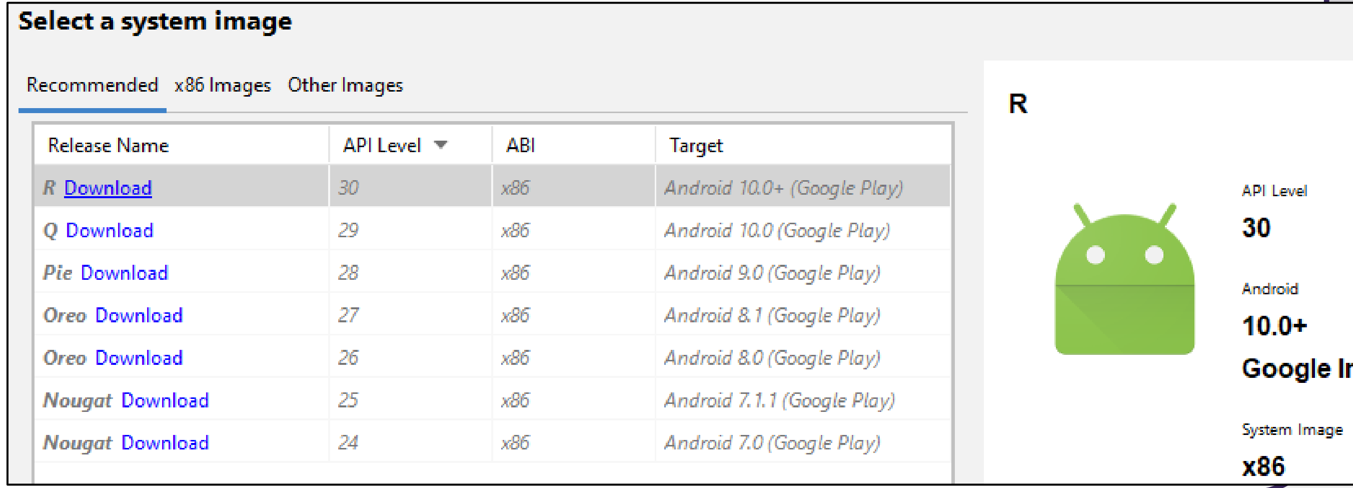
Choose a device definition

Category
TV
Phone
Wear OS
Tablet
Automotive

Name	Play Store	Size	Resolution	Density
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

Nexus 5X

Size: large
Ratio: long
Density: 420dpi



Select a system image

Release Name	API Level	ABI	Target
R Download	30	x86	Android 10.0+ (Google Play)
Q Download	29	x86	Android 10.0 (Google Play)
Pie Download	28	x86	Android 9.0 (Google Play)
Oreo Download	27	x86	Android 8.1 (Google Play)
Oreo Download	26	x86	Android 8.0 (Google Play)
Nougat Download	25	x86	Android 7.1.1 (Google Play)
Nougat Download	24	x86	Android 7.0 (Google Play)

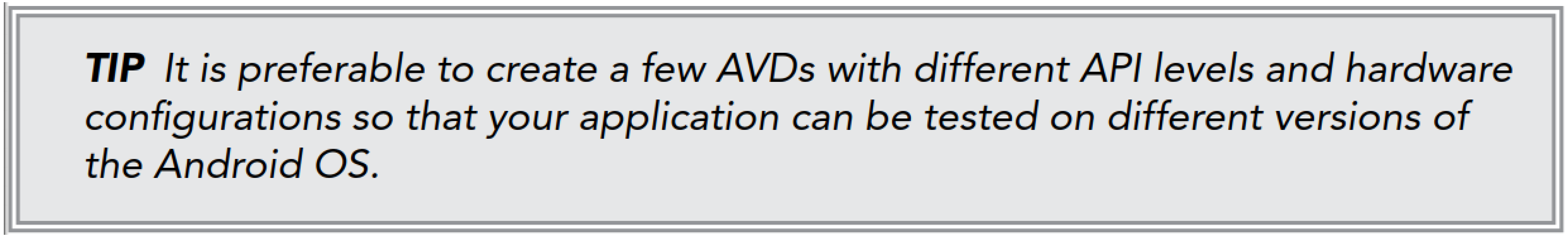
R

API Level
30

Android
10.0+
Google In

System Image
x86

Creating Android Virtual Devices (AVDs)


TIP *It is preferable to create a few AVDs with different API levels and hardware configurations so that your application can be tested on different versions of the Android OS.*

Testing Android Studio Apps on a Physical Android Device

- While 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
- Communication with both AVD instances and connected Android devices is handled by the Android Debug Bridge (ADB).
- ADB facilitates interaction between 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 background process running in either AVDs or real Android devices.
- The ADB client is provided in the form of a command-line tool named adb located in the Android SDK platform-tools sub-directory and Android Studio also has a built-in client.

Android Debug Bridge (ADB) Tool

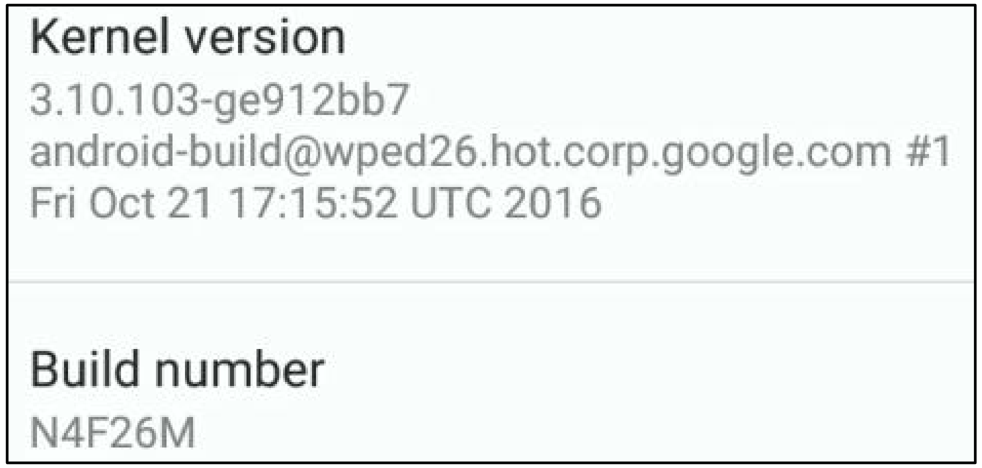
- A variety of tasks may be performed using the adb command-line tool. For example, a listing of currently active virtual or physical devices.
- The following command output indicates the presence of an AVD on the system but no physical devices:

```
$ adb devices
List of devices attached
emulator-5554   device
```

Enabling ADB on Android based Devices

- Before ADB can connect to an Android device, that device must first be configured to allow the connection.

- On phone and tablet devices running Android 6.0 or later, following steps can be followed:
    1. Open the Settings app on the device and select the About tablet or About phone option.
    2. On the About screen, scroll down to the Build number field and tap on it seven times until a message appears indicating that developer mode has been enabled.
    3. Enable the USB Debugging from Developer Options



Kernel version
3.10.103-ge912bb7
android-build@wped26.hot.corp.google.com #1
Fri Oct 21 17:15:52 UTC 2016

Build number
N4F26M

Windows ADB Configuration

- Make sure that the Android SDK platform-tools directory is included in the operating system PATH environment variable
- Install the appropriate USB drivers for the physical device on the system
- With the drivers installed and the device now being recognized as the correct device type, open a Command Prompt window and execute the following command:

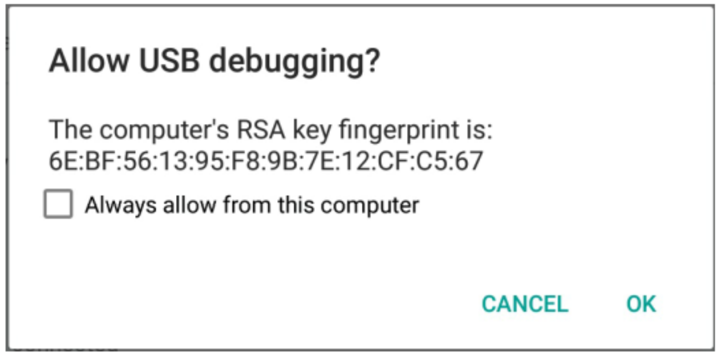
```
adb devices
```

- This command should output information about the connected device similar to the following:

```
List of devices attached
HT4CTJT01906        offline
```

Windows ADB Configuration (Issues)

- If the device is listed as offline or unauthorized, go to the device display and check for permission to Allow USB debugging.



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

- In case that the device is not listed, restart the ADB server by executing the command:

```
adb kill-server
adb start-server
```

- If the device is still not listed, try executing the following command:

```
android update adb
```

- Note that it may also be necessary to reboot the system.

Configure Your Project

Basic Configuration

- After selecting to create new project you need to describe your application by filling out some fields:

1. Application Name - This name will be shown to the user.

Example: Hello World. You can always change it later in AndroidManifest.xml file.

2. Company Domain - This is the qualifier for your project's package name.

Example: stackoverflow.com.

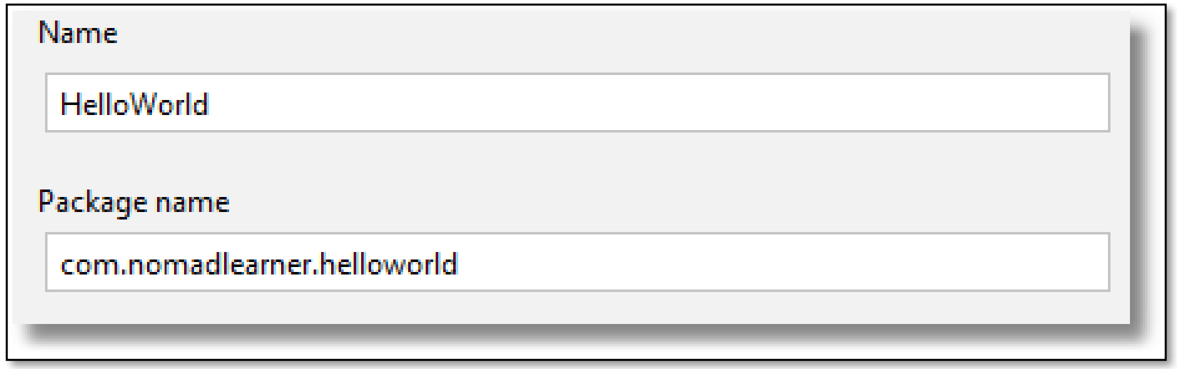
3. Package Name (aka applicationId) - This is the fully qualified project package name.

4. Project Location - This is the directory where your project will be stored.

Selecting the Package Name

- It should follow Reverse Domain Name Notation (aka Reverse DNS): Top Level Domain . Company Domain . [Company Segment .] Application Name.

- Example: com.nomadlearner.android.helloworld or com. nomadlearner.helloworld. You can always change your applicationId by overriding it in your gradle **fi**le.



Name	HelloWorld
Package name	com.nomadlearner.helloworld

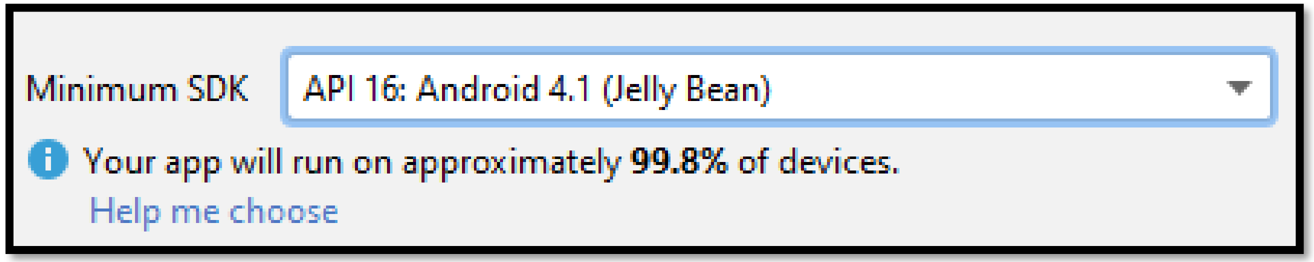
Don't use the default pre**fi**x "com.example" unless you don't intend to submit your application to the Google Play Store.

The package name will be your unique applicationId in Google Play.

Select API Level

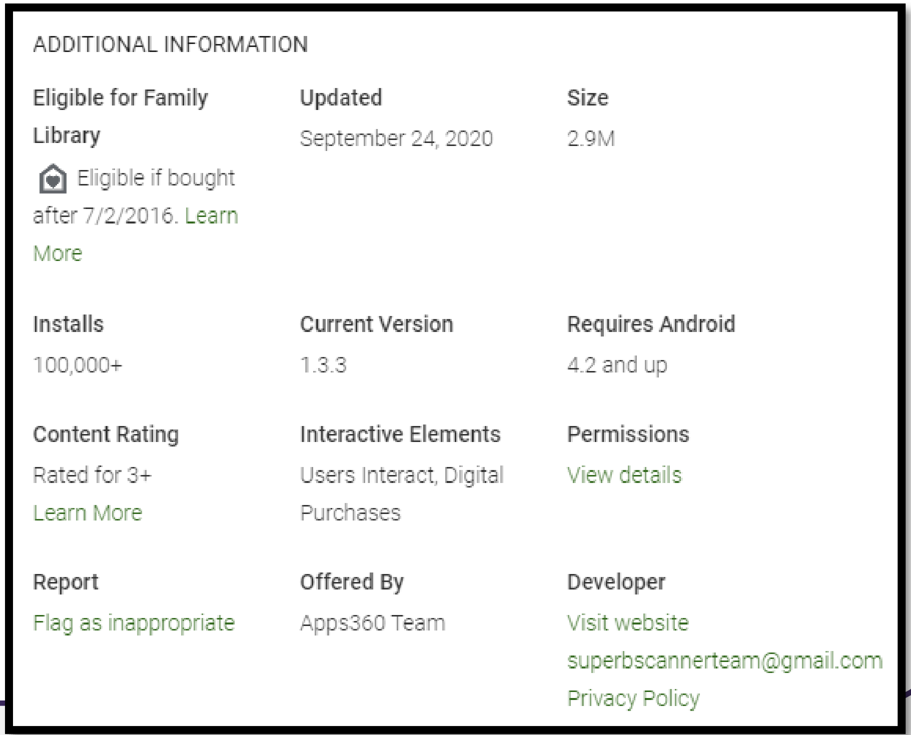
- The Minimum SDK is the lower bound for your app.
- It is one of the signals the Google Play Store uses to determine which devices an app can be installed on

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



Minimum SDK: API 16: Android 4.1 (Jelly Bean)

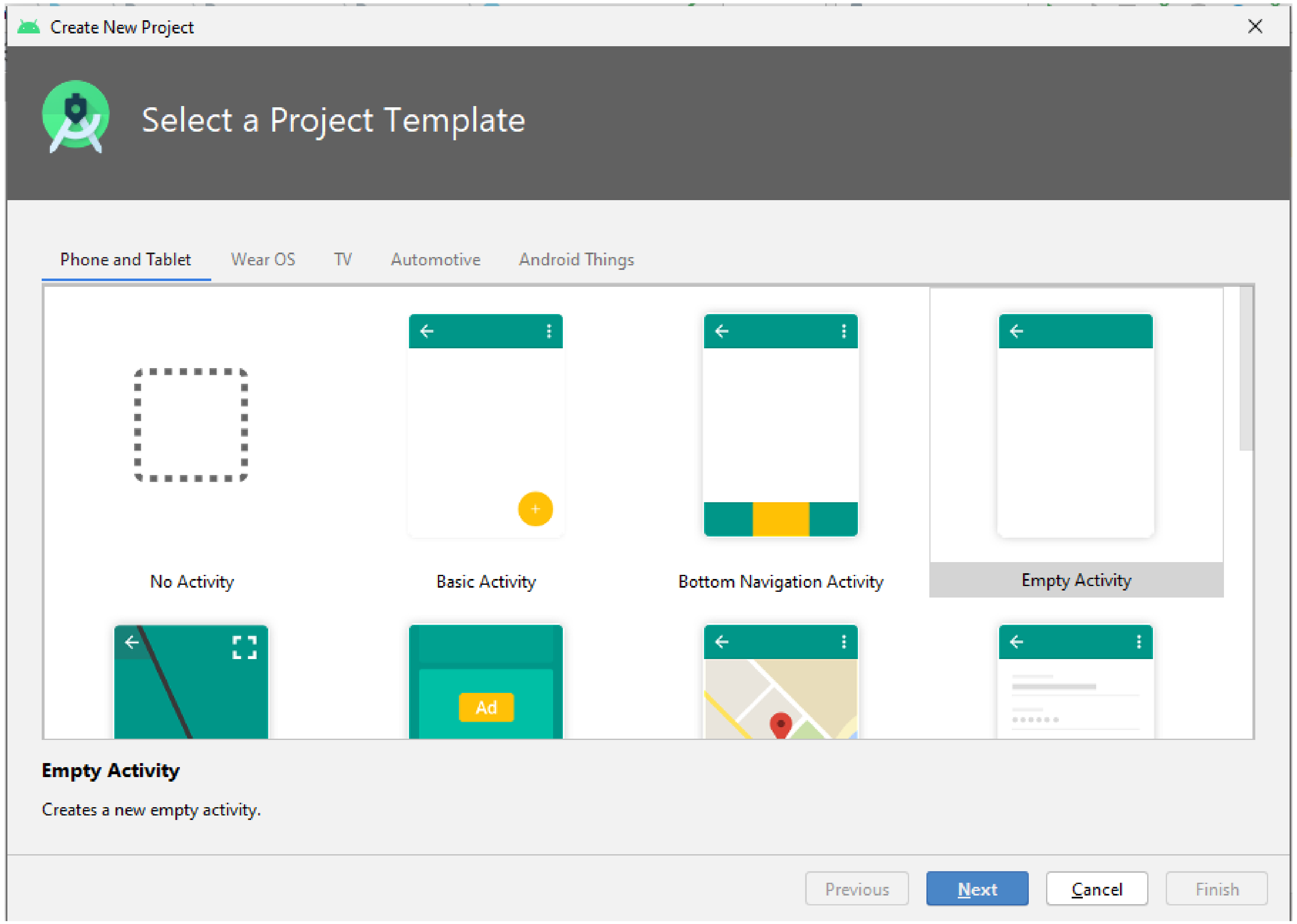
Your app will run on approximately **99.8%** of devices.
Help me choose



ADDITIONAL INFORMATION

Eligible for Family Library	**Updated**	**Size**
Eligible if bought after 7/2/2016. Learn More	September 24, 2020	2.9M
Installs	**Current Version**	**Requires Android**
100,000+	1.3.3	4.2 and up
Content Rating	**Interactive Elements**	**Permissions**
Rated for 3+ Learn More	Users Interact, Digital Purchases	View details
Report	**Offered By**	**Developer**
Flag as inappropriate	Apps360 Team	Visit website superbscannerteam@gmail.com Privacy Policy

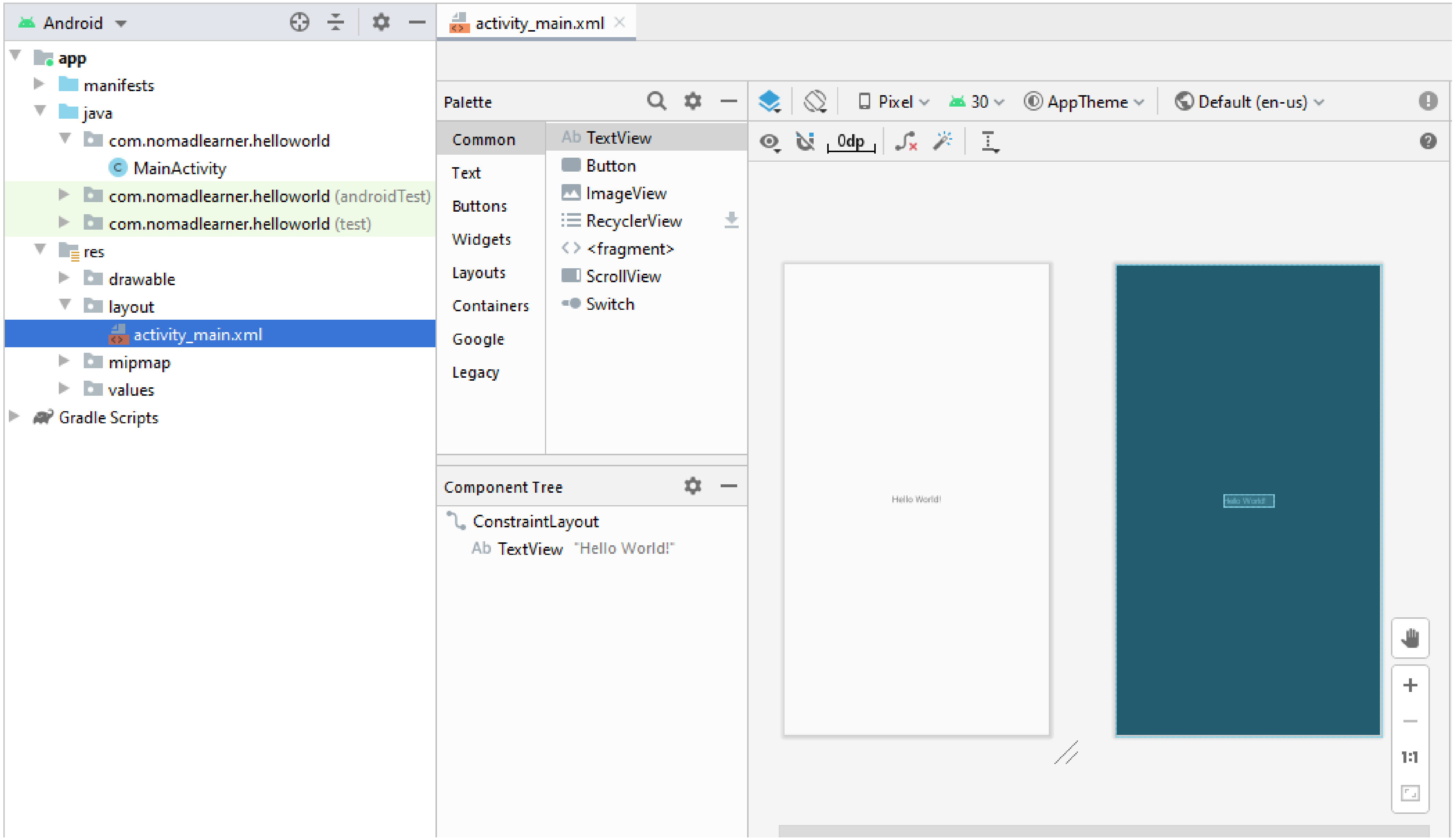
Choosing Project Template



Empty Activity

Creates a new empty activity.

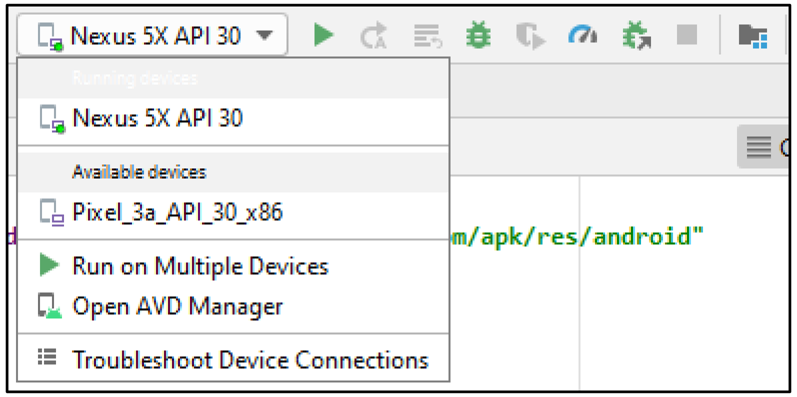
A Simple Hello Word Project



Naming Convention

- It is accepted practice in Android development to name your main activity—that is, the activity that is loaded on startup by your application—as MainActivity.
- The reason for this is to make it easier to locate the startup code (starting point) for your application.
- All other activities can be named by their function, for example InputFormActivity or DeleteRecordActivity.
- The layout file follows the “name” naming convention. The startup layout is the activity_main layout.
- All other layouts should be named according to the activity that they support (activity_input, activity_delete).

Launching your First Android App

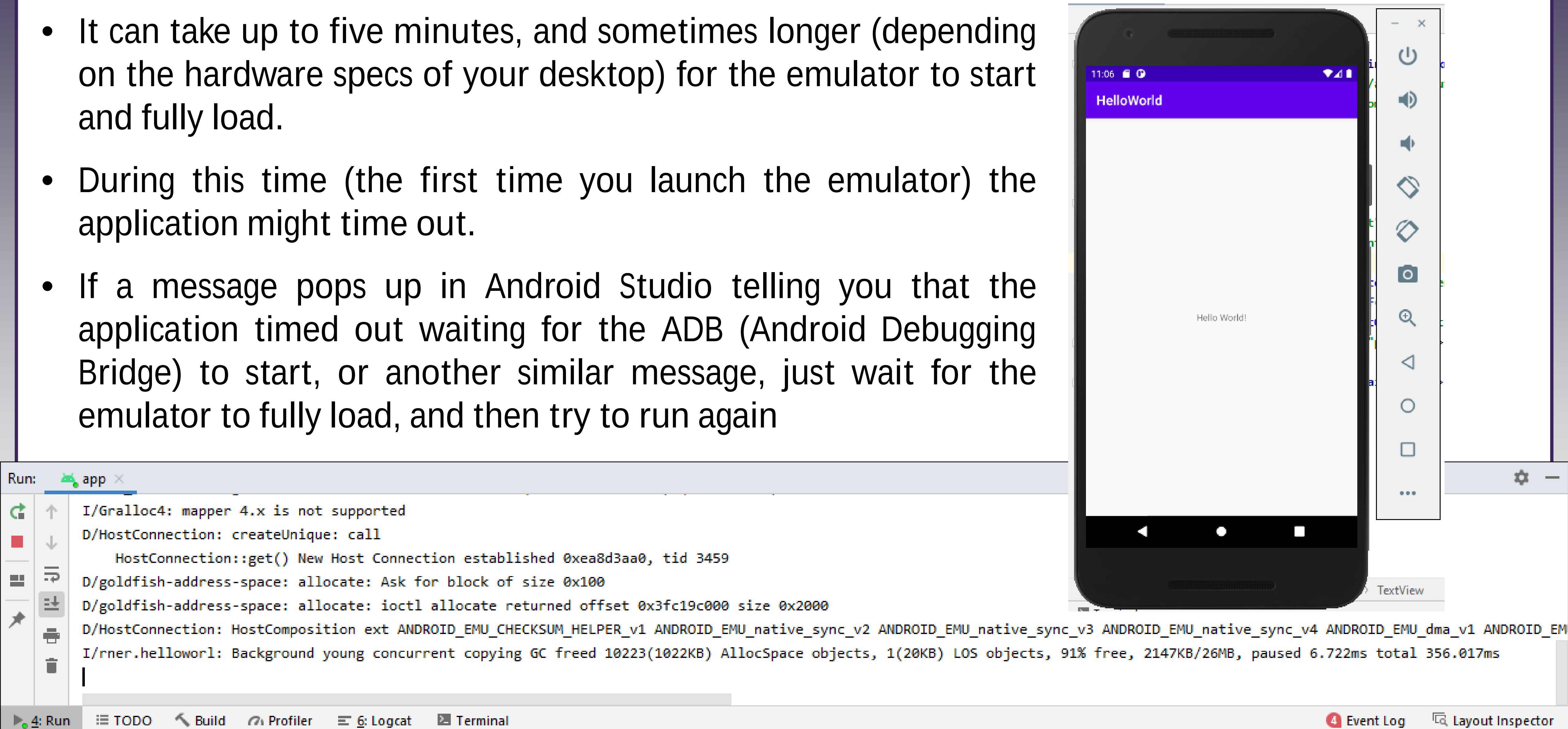


```
m/apk/res/android"
```

NOTE *Note that If there's ever a time when you have not already created the emulator, you can create an emulator at this point.*

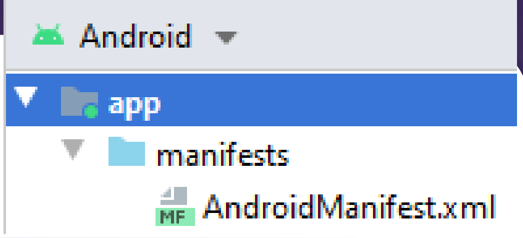
Launching your First Android App

- It can take up to five minutes, and sometimes longer (depending on the hardware specs of your desktop) for the emulator to start and fully load.
- During this time (the first time you launch the emulator) the application might time out.
- If a message pops up in Android Studio telling you that the application timed out waiting for the ADB (Android Debugging Bridge) to start, or another similar message, just wait for the emulator to fully load, and then try to run again



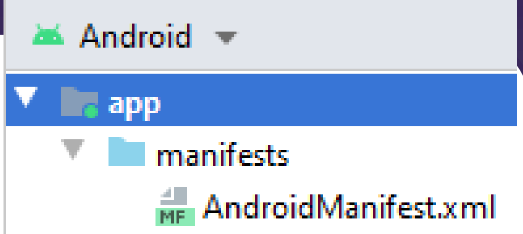
```
I/Gralloc4: mapper 4.x is not supported
D/HostConnection: createUnique: call
    HostConnection::get() New Host Connection established 0xea8d3aa0, tid 3459
D/goldfish-address-space: allocate: Ask for block of size 0x100
D/goldfish-address-space: allocate: ioctl allocate returned offset 0x3fc19c000 size 0x2000
D/HostConnection: HostComposition ext ANDROID_EMU_CHECKSUM_HELPER_v1 ANDROID_EMU_native_sync_v2 ANDROID_EMU_native_sync_v3 ANDROID_EMU_native_sync_v4 ANDROID_EMU_dma_v1 ANDROID_EM
I/rner.helloworl: Background young concurrent copying GC freed 10223(1022KB) AllocSpace objects, 1(20KB) LOS objects, 91% free, 2147KB/26MB, paused 6.722ms total 356.017ms
```

App Manifest

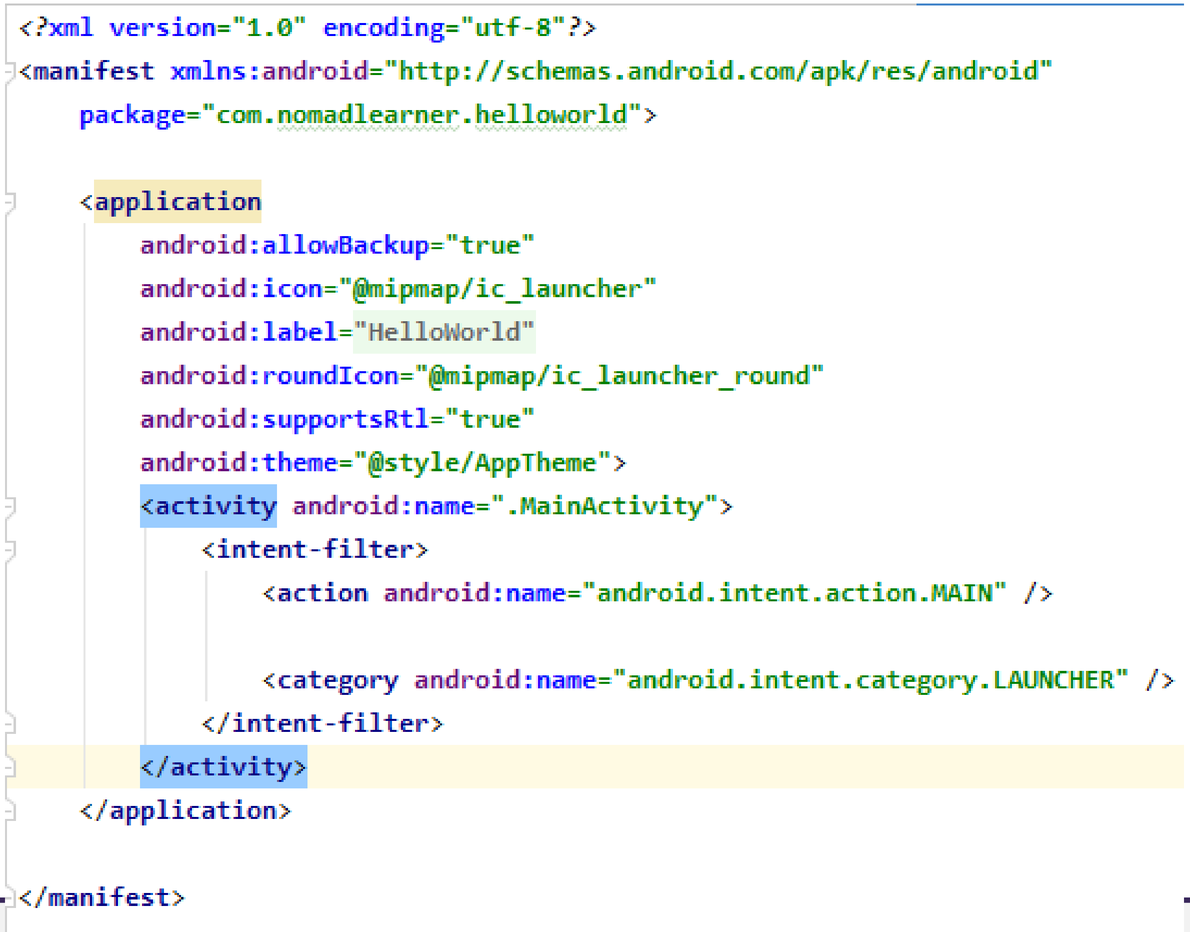


- The Android manifest file describes some of the basic information about an Android application.
- Every app project must have an AndroidManifest.xml file (with precisely that name) at the root of the project source set.
- It contains the declaration of our activities, as well as some more advanced components.
- The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Play.
- manifest file is required to declare the following:
  1. Package name and application ID
  2. App components
  3. Permissions
  4. Device compatibility

App Manifest



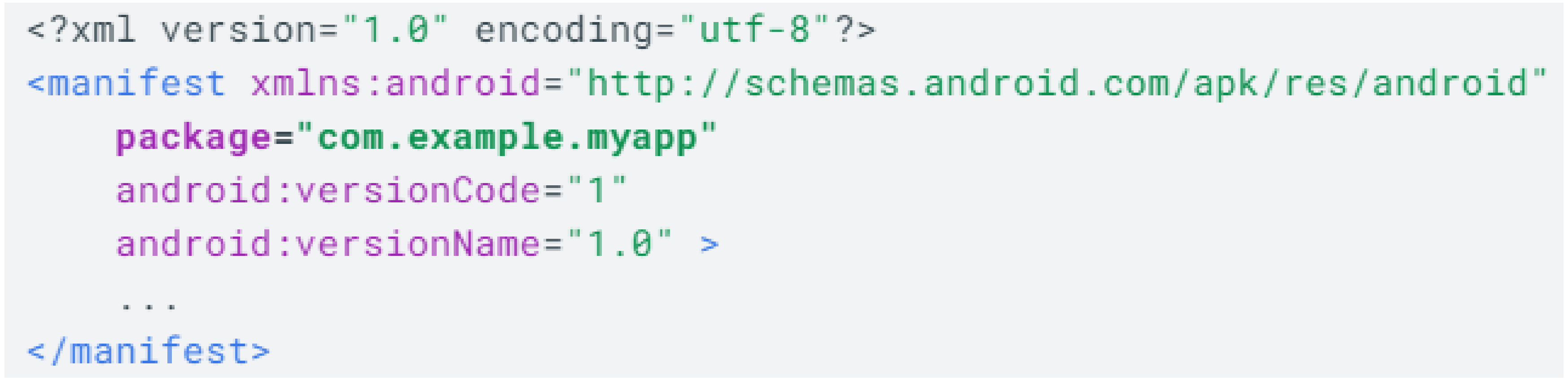
- If you're using Android Studio to build your app, the manifest file is created for you, and most of the essential manifest elements are added as you build your app (especially when using code templates).



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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HelloWorld"
        android:roundIcon="@mipmap/ic_launcher_round"
        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>
```

1. Package name and application ID

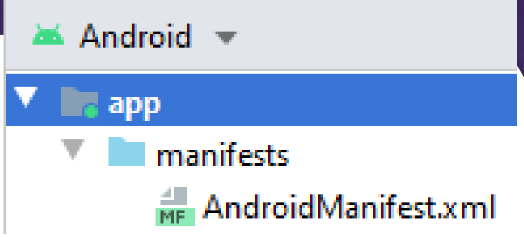
- The manifest file's root element requires an attribute for your app's package name (usually matching your project directory structure—the Java namespace).



```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp"
    android:versionCode="1"
    android:versionName="1.0" >
    ...
</manifest>
```

The root <manifest> element with the package name "com.example.myapp":

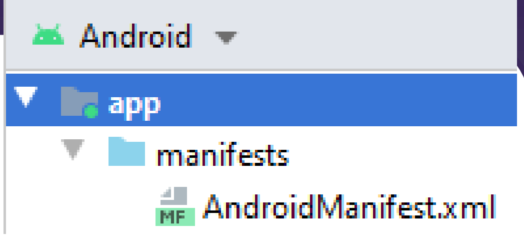
- Once the APK is compiled, the package attribute also represents your app's universally unique application ID.



2. App Components

- For each app component that you create in your app, you must declare a corresponding XML element in the manifest file:
    - `<activity>` for each subclass of `Activity`.
    - `<service>` for each subclass of `Service`.
    - `<receiver>` for each subclass of `BroadcastReceiver`.
    - `<provider>` for each subclass of `ContentProvider`.
- If you subclass any of these components without declaring it in the manifest file, the system cannot start it.

2. App Components



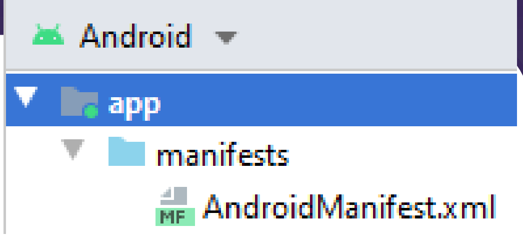
- The name of your subclass must be specified with the name attribute, using the full package designation. For example, an Activity subclass can be declared as follows:

```xml
<manifest ... >
    <application ... >
        <activity android:name="com.example.myapp.MainActivity" ... >
        </activity>
    </application>
</manifest>
```

- However, if the first character in the name value is a period, the app's package name is prefixed to the name. For example, the following activity name is resolved to "com.example.myapp.MainActivity":

```xml
<manifest package="com.example.myapp" ... >
    <application ... >
        <activity android:name=".MainActivity" ... >
            ...
        </activity>
    </application>
</manifest>
```

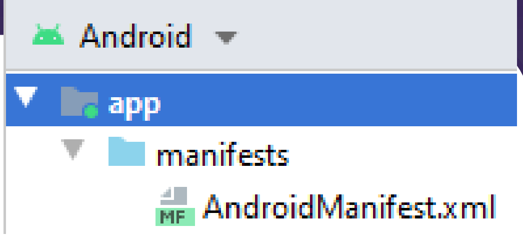
3. Permissions



- Android apps must request permission to access sensitive user data (such as contacts and SMS) or certain system features (such as the camera and internet access).
- Each permission is identified by a unique label.
- For example, an app that needs to send SMS messages must have the following line in the manifest:

```
<manifest ... >
    <uses-permission android:name="android.permission.SEND_SMS"/>
    ...
</manifest>
```

- Beginning with Android 6.0 (API level 23), the user can approve or reject some app permissions at runtime. But no matter which Android version your app supports, you must declare all permission requests with `<uses-permission>` element in the manifest



4. Device Compatibility

- In the manifest file you can declare what types of hardware or software features your app requires, and thus, which types of devices your app is compatible with.
- Google Play Store does not allow your app to be installed on devices that don't provide the features or system version that your app requires.
- There are several manifest tags that define which devices your app is compatible with. For example <uses-feature>, <uses-sdk> etc
- The `<uses-feature>` element allows you to declare hardware and software features your app needs. For example, if your app needs to use compass sensor

```
<manifest ... >
    <uses-feature android:name="android.hardware.sensor.compass"
                  android:required="true" />
    ...
</manifest>
```

Read More about Android Studio and SDK

- You can read more about Android SDK in the following article online:

https://medium.com/@92alanc/android-basics-2-android-studio-jdk-and-sdk-a16241916b6d#:~:text=Android%20Studio%20is%20the%20official%20IDE%20for%20Android%20development.&text=They%20perform%20a%20lot%20better,You%20can%20download%20it%20here.

Recommended Readings

- Page # 14 to 26, Chapter # 01: Getting Started with Android Programming from Beginning Android Programming with Android Studio, 4th Edition by J. F. DiMarzio, Wrox, 2017

- Page # 02 to 12, Chapter 1: Getting started with Android from, AndroidNotesForProfessionals by GoalKicker.com

- User Guide: developer.android.com/studio/intro