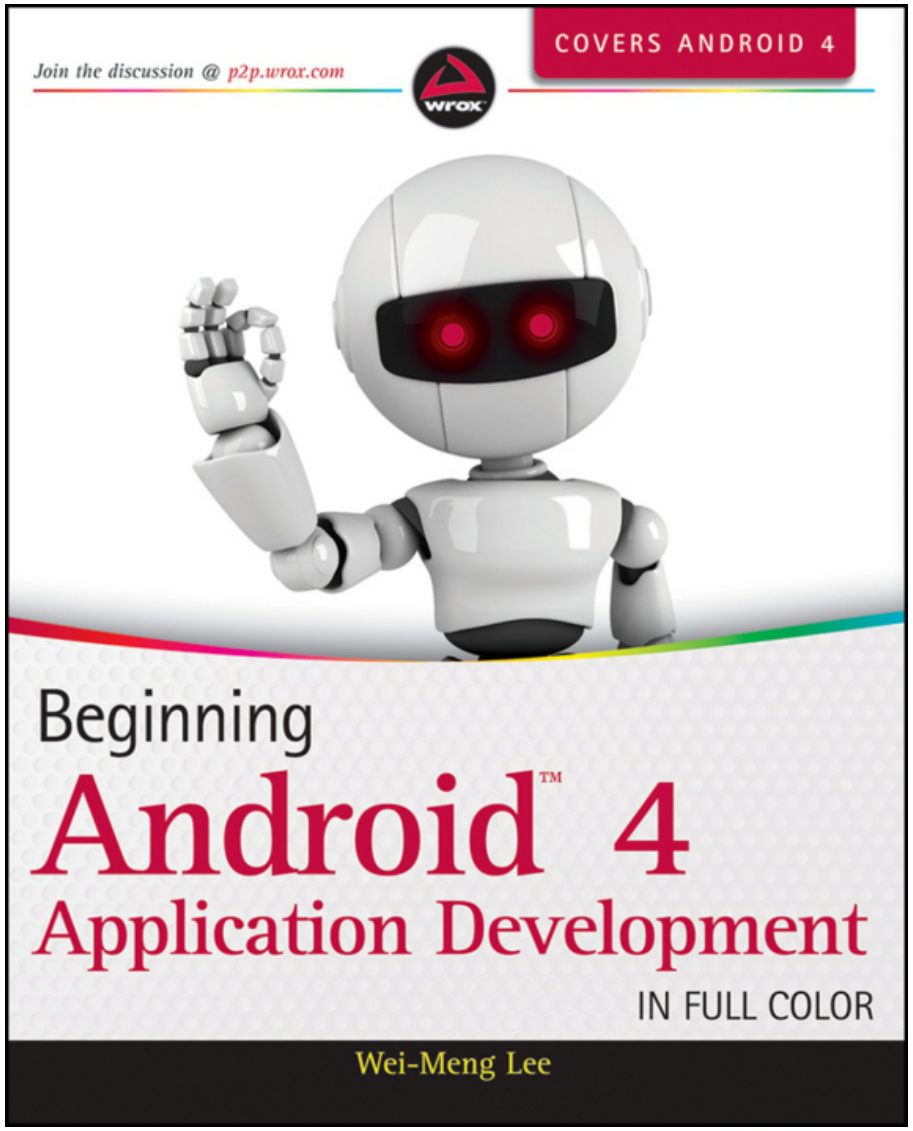
Android Development

Chapter 1 – Getting Started with Android Programming

katholieke hogeschool
associatie KU Leuven
vives

Beginning Android 4 Application Development

- Amazon.co.uk
  - http://www.amazon.co.uk/Beginning-Android-4-Application-Development/dp/1118199545/ref=pd_sim_b_2?ie=UTF8&refRID=0HCW1DQQPYNSVQ3A53D8
- Bol.com
  - http://www.bol.com/nl/p/beginning-android-4-application-development/1001004011690816/



Android Development
Chapter 1 – Getting Started with Android Programming

Android and its history

katholieke hogeschool
associatie KU Leuven
vives

Android – The Mobile Operating System

- Android is a mobile operating system
  - Based on the Linux kernel (version 2.6)
- Originally developed by a startup company called Android inc.
- Bought by Google in 2005
- Released under the open source Apache License

- Manufacturer's such as Motorola and Sony Ericsson saw Android as a replacement for their own mobile OS's
- Main advantage is that Android offers a unified approach to application development
  - Developers only need to develop for Android and their app should run on numerous Android devices

Android Devices

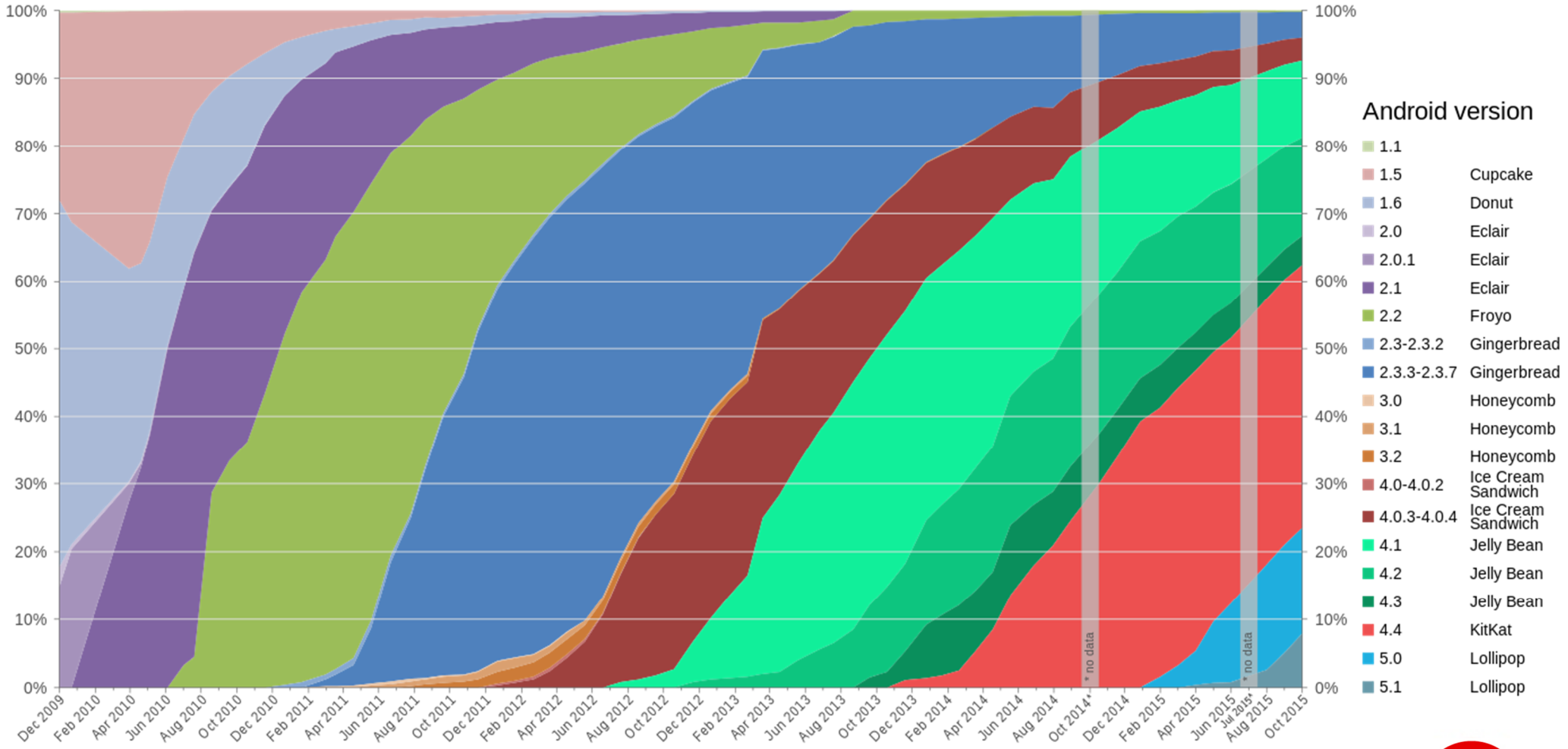
- Smartphones
- Tablets
- e-book readers
- Watches
- TVs

katholieke hogeschool
associatie KU Leuven
vives

Android Versions

Code name	Version	API level
(no code name)	1.0	API level 1
(no code name)	1.1	API level 2
Cupcake	1.5	API level 3, NDK 1
Donut	1.6	API level 4, NDK 2
Eclair	2.0	API level 5
Eclair	2.0.1	API level 6
Eclair	2.1	API level 7, NDK 3
Froyo	2.2.x	API level 8, NDK 4
Gingerbread	2.3 - 2.3.2	API level 9, NDK 5
Gingerbread	2.3.3 - 2.3.7	API level 10
Honeycomb	3.0	API level 11
Honeycomb	3.1	API level 12, NDK 6
Honeycomb	3.2.x	API level 13
Ice Cream Sandwich	4.0.1 - 4.0.2	API level 14, NDK 7
Ice Cream Sandwich	4.0.3 - 4.0.4	API level 15, NDK 8
Jelly Bean	4.1.x	API level 16
Jelly Bean	4.2.x	API level 17
Jelly Bean	4.3.x	API level 18
KitKat	4.4 - 4.4.4	API level 19
Lollipop	5.0	API level 21

Android Versions



katholieke hogeschool
associatie KU Leuven
vives

Android Versions - Milestones

- Android 1.6, Donut
  - The world's information is at your fingertips – search the web, get driving directions ... or just watch cat videos.
- Android 2.0, Eclair
  - Make your home screen just how you want it. Arrange apps and widgets across multiple screens and in folders. Stunning live wallpapers respond to your touch.
- Android 2.2, Froyo
  - Voice Typing lets you input text, and Voice Actions let you control your phone, just by speaking.
- Android 2.3, Gingerbread
  - New sensors make Android great for gaming - so you can touch, tap, tilt, and play away.

Android Versions - Milestones

- Android 3.0, Honeycomb
  - Optimized for tablets
  - New UI, refined multitasking, tabbed browsing, auto fill form, bookmark synchronization, private browsing, ...
  - New features are not supported on smartphones

- Android 4.0, Ice Cream Sandwich
  - Android comes of age with a new, refined design. Simple, beautiful and beyond smart.
  - All 3.0 features are supported on smartphones as well

Android Versions - Milestones

- Android 4.1, Jelly Bean
  - Android is fast and smooth with buttery graphics. With Google Now, you get just the right information at the right time.
  - With more than 1 million apps on Google Play, and thousands of Android devices, you've got the freedom to do what you want on any device you choose.

- Android 4.4, KitKat
  - Smart, simple, and truly yours. A more polished design, improved performance, and new features.
  - New features introduced such as facial recognition unlock, data usage monitoring and control, Near Field Communication (NFC)

Android Versions - Milestones

- Android 5.0, Lollipop
  - A sweet new take on Android. Get the smarts of Android on screens big and small – with the right information at the right moment.
  - Notification control, better battery management, security, device sharing, support for TVs, ...
  - Released November the 3th (2014)

- Android 5.1, Lollipop
  - Ability to join Wi-Fi networks and control paired Bluetooth devices from quick settings
  - Official support for multiple SIM cards
  - Device protection: if a device is lost or stolen it will remain locked until the owner signs into their Google account, even if the device is reset to factory settings.
  - High-definition voice calls, available between compatible devices running Android 5.1

Android Versions - Milestones

- Android 6.0, Marshmallow
  - Unveiled under the codename "Android M" during Google I/O on May 28, 2015
  - For the Nexus 5 and Nexus 6 phones, Nexus 9 tablet, and Nexus Player set-top box
  - Native fingerprint reader support
  - Direct Share feature for target-specific sharing between apps
  - Do Not Disturb mode
  - Post-install/run-time permission requests
  - USB Type-C support
  - Automatic full data backup and restore for apps
  - 4K Display mode for apps
  - Adoptable External storage to behave like Internal Storage
  - MIDI support for musical instruments
  - Experimental Multi Window feature

Android Features

- Because Android is open source and freely available to manufacturers for customization, there are no fixed hardware or software configurations. However, Android itself supports the following features:

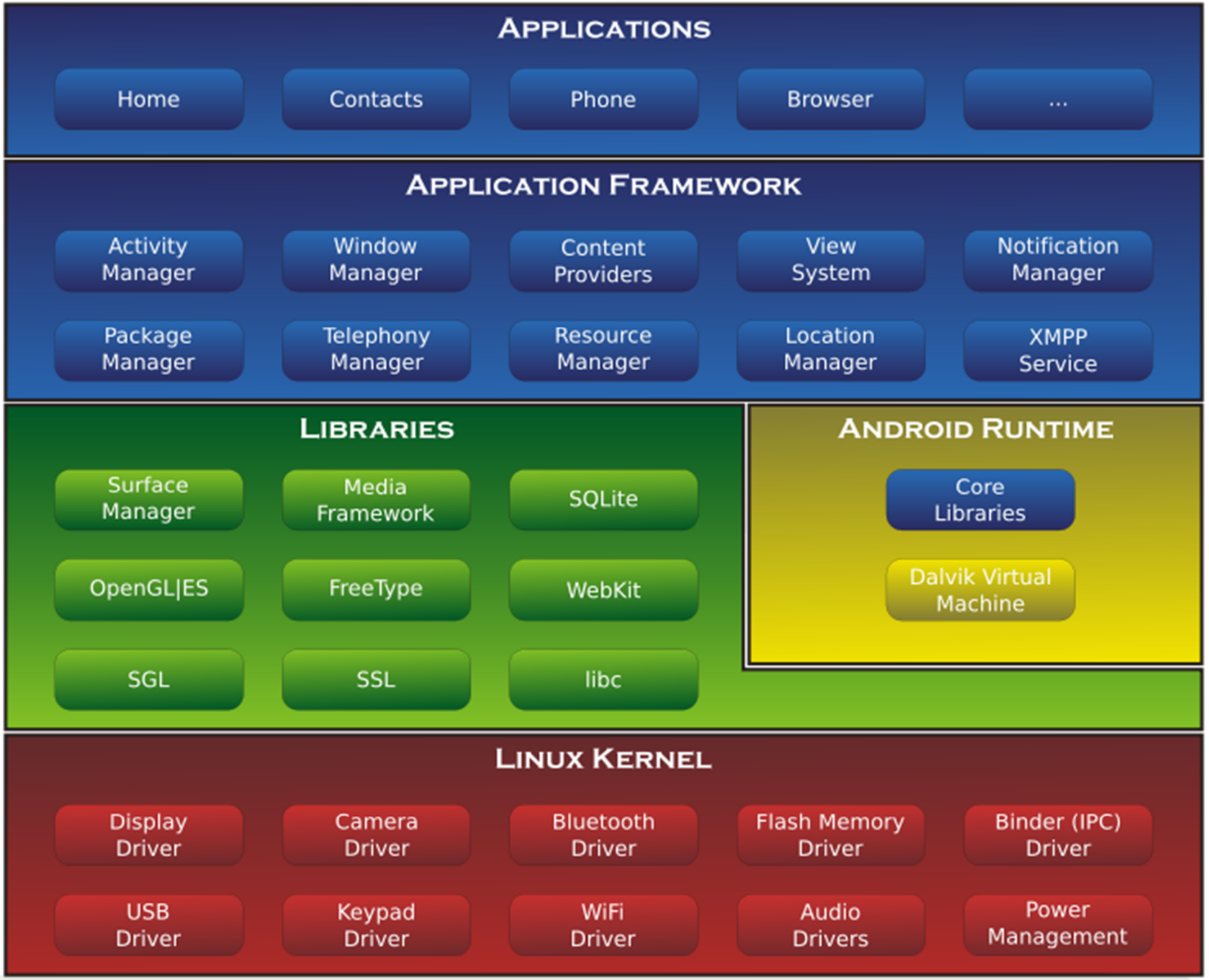
- **Storage** - Uses SQLite, a lightweight relational database, for data storage.
- **Connectivity** - Supports GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (includes A2DP and AVRCP), Wi-Fi, LTE, and WiMAX.
- **Messaging** - Supports both SMS and MMS.
- **Web browser** - Based on the open source WebKit, together with Chrome's V8 JavaScript engine
- **Media support** - Includes support for the following media: H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP
- **Hardware support** - Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor, and GPS
- **Multi-touch** - Supports multi-touch screens
- **Multi-tasking** - Supports multi-tasking applications
- **Flash support** - Android 2.3 supports Flash 10.1.
- **Tethering** - Supports sharing of Internet connections as a wired/wireless hotspot

Android Development
Chapter 1 – Getting Started with Android Programming

Android Architecture

katholieke hogeschool
associatie KU Leuven
vives

Android Architecture



Linux Kernel

- Android is build on top of the Linux 2.6 kernel
  - Linux core functionality
    - Memory management
    - Process management
    - Networking
    - Security settings
  - Low-level device drivers for various hardware components of an Android device

Libraries

- Contain all the code that provides the main features of an Android OS.
- Examples:
  - SQLite library provides database support so that applications can use it for data storage
  - WebKit library provides functionalities for web browsing
  - Libc: c standard lib.
  - SSL: Secure Socket Layer
  - SGL: 2D image engine
  - OpenGL|ES: 3D image engine
  - Media Framework: media codecs
  - FreeType: Bitmap and Vector
  - SurfaceManager: Compose window manager with off-screen buffering

Android Runtime

- Core Libraries
  - Provides the functionality of the JAVA Programming Language

- Dalvik VM
  - A type of Java Virtual Machine
  - Register based (not stack machine like JVM)
  - Optimization for low memory requirements
  - Executes .dex (Dalvik-Executable) files instead of .class
    - Android application (APK) files contain executable bytecode files in the form of Dalvik Executable (DEX) files, which contain the compiled code used to run your app.
  - DX tool converts classes to .dex format

- Each Android application:
  - runs on its own Process
  - runs on its own Instance of Dalvik VM
  - is assigned its own Linux user ID

Android Runtime

- As of Android 5.0 (Lollipop) Google made internal changes to the platform, with the Android Runtime (ART) officially replacing Dalvik
  - Improved application performance
  - Changes intended to improve and optimize battery usage, known internally as Project Volta.

- Dalvik
  - Android apps are deployed in Dalvik bytecode, which is portable, unlike native code.
  - To run the app on a device, the code has to be compiled to machine code.
  - Dalvik is based on JIT (just in time) compilation.
    - Each time you run an app, the part of the code required for its execution is going to be translated (compiled) to machine code at that moment.
    - As you progress through the app, additional code is going to be compiled and cached, so that the system can reuse the code while the app is running.
  - Pro: Smaller memory footprint and uses less physical space

Android Runtime

- ART (Android RunTime)
  - Compiles the intermediate language, Dalvik bytecode, into a system-dependent binary.
  - The whole code of the app will be pre-compiled during install (once), thus removing the lag that we see when we open an app on our device.
  - With no need for JIT compilation, the code should execute much faster.

- Except for the potential speed increase, the use of ART can provide an important secondary benefit.
  - As ART runs app machine code directly (native execution), it doesn't hit the CPU as hard as just-in-time code compiling on Dalvik.
  - Less CPU usage results in less battery drain, which is a big plus for portable devices in general.

Application Framework

- Exposes the various capabilities of the Android OS to application developers so they can be used in their apps

- Important blocks:
  - **Activity Manager:** Manages the activity life cycle of applications
  - **Content Providers:** Manage the data sharing between applications
  - **Telephony Manager:** Manages all voice calls. We use telephony manager if we want to access voice calls in our application.
  - **Location Manager:** Location management, using GPS or cell tower
  - **Resource Manager:** Manage the various types of resources we use in our Application

Applications

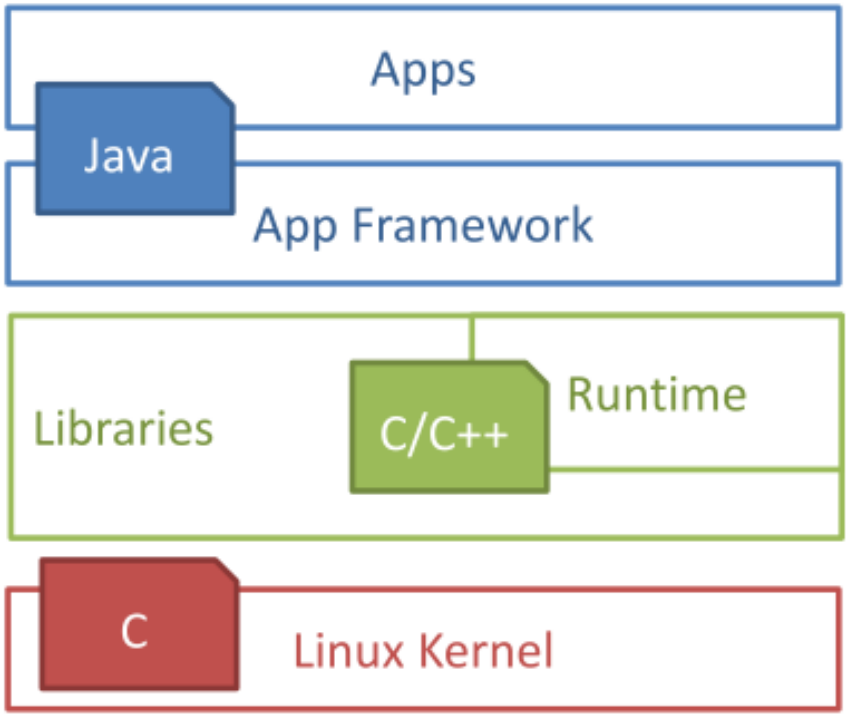
- This is where our applications are placed.

- Some pre-installed applications:
  - SMS client app
  - Dialer
  - Web browser
  - Contact manager
  - ...

- As developers, we are able to write an app which replaces any existing system app.
  - No compulsory applications
  - Equality among apps
  - Easily embedded web browser
  - Parallel running

Used languages

- Design goals
  - Open Source
  - High flexibility
  - High data accessibility
  - Rapid development (XML, Java)
- Used Languages
  - App: Java
  - Framework: Java
  - Libraries: C/C++
  - OS & Drivers: C



Android Development
Chapter 1 – Getting Started with Android Programming

Developing for Android

katholieke hogeschool
associatie KU Leuven
vives

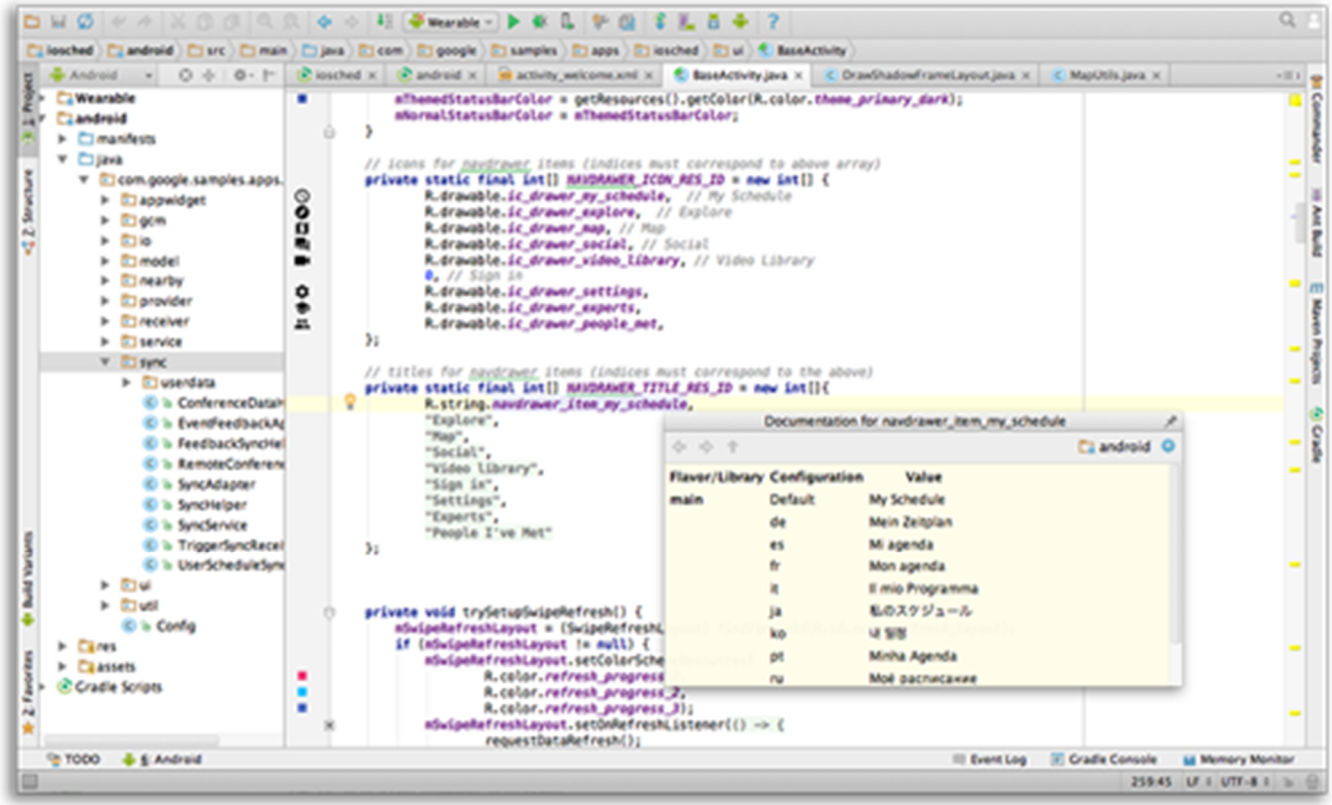
Android Studio

- Android Studio is an integrated development environment (IDE) for developing for the Android platform.
  - Freely available under the Apache License 2.0.
  - First stable build was released in December 2014

- Android Studio is designed specifically for Android development.
- It is available for download on Windows, Mac OS X and Linux
- Replaces Eclipse Android Development Tools (ADT) as Google's primary IDE for native Android application development.

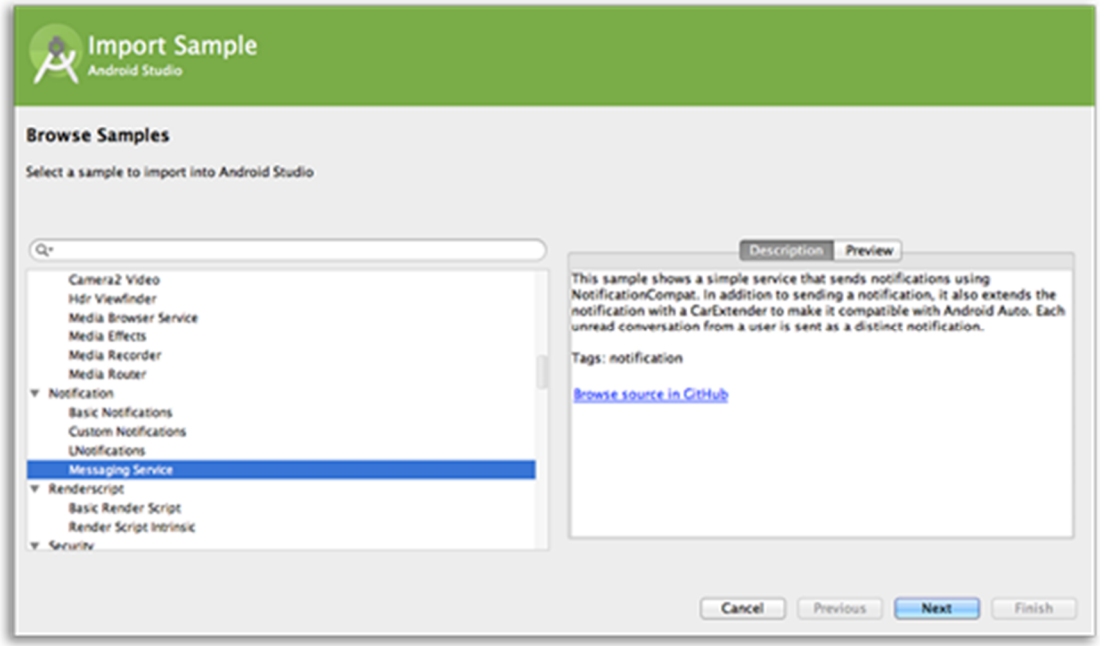
Android Studio

- Intelligent code editor
  - At the core of Android Studio is an intelligent code editor capable of advanced code completion, refactoring, and code analysis.



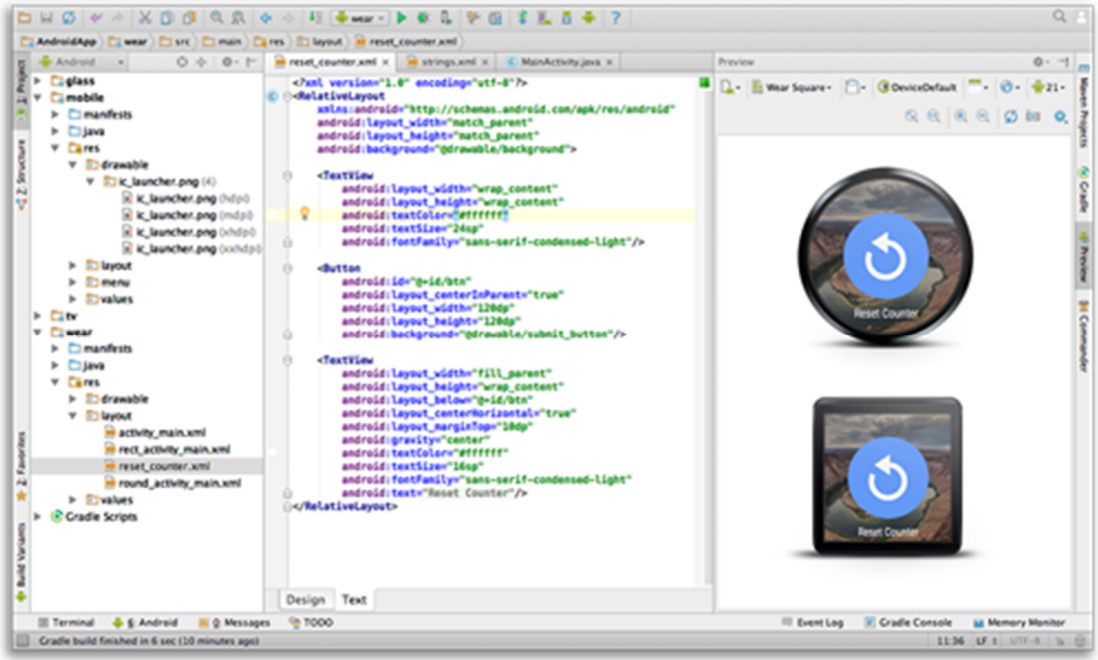
Android Studio

- Code templates and GitHub integration
  - New project wizards make it easier than ever to start a new project.
  - Start projects using template code for patterns such as navigation drawer and view pagers, and even import Google code samples from GitHub.



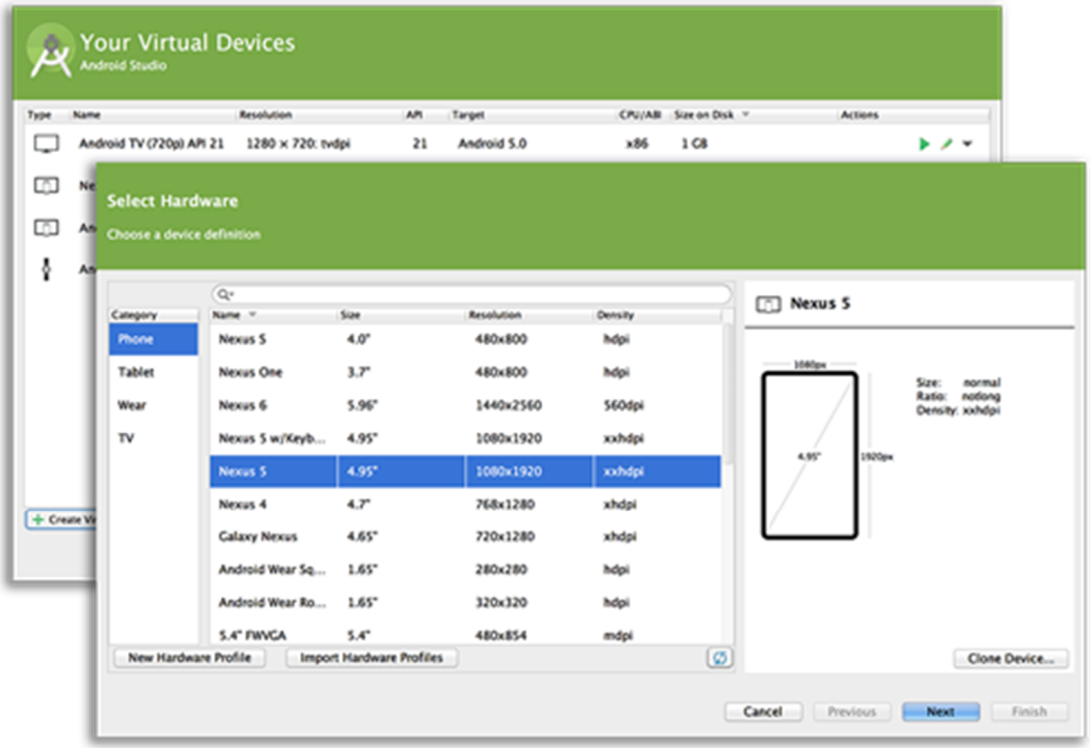
Android Studio

- Multi-screen app development
  - Build apps for Android phones, tablets, Android Wear, Android TV, Android Auto and Google Glass.
  - With the new Android Project View and module support in Android Studio, it's easier to manage app projects and resources.



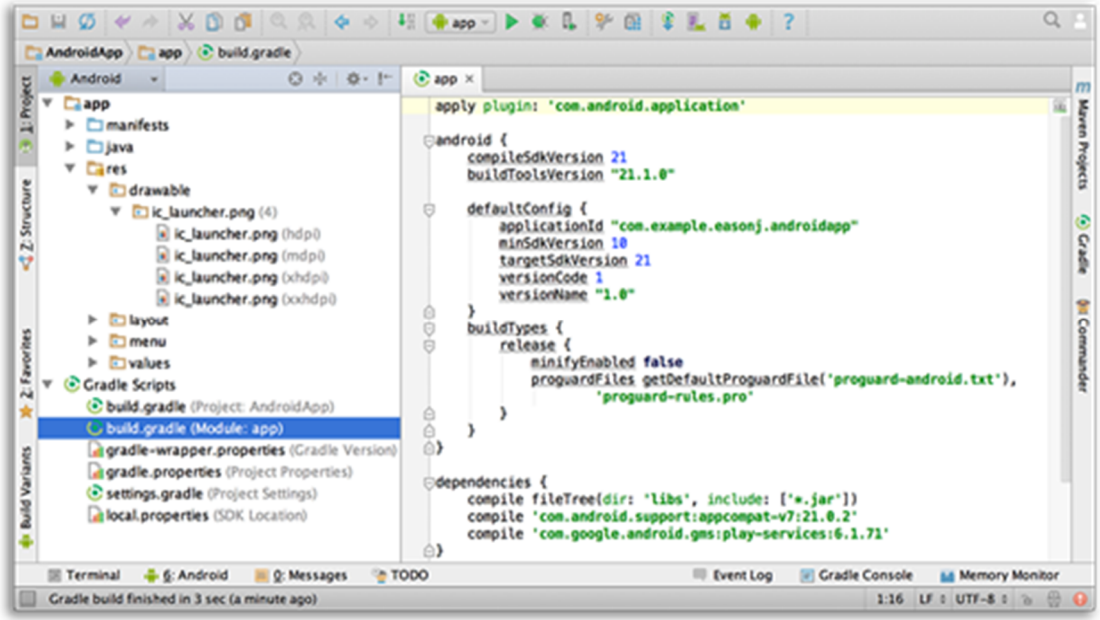
Android Studio

- Virtual devices for all shapes and sizes
  - Android Studio comes pre-configured with an optimized emulator image.
  - The updated and streamlined Virtual Device Manager provides pre-defined device profiles for common Android devices.

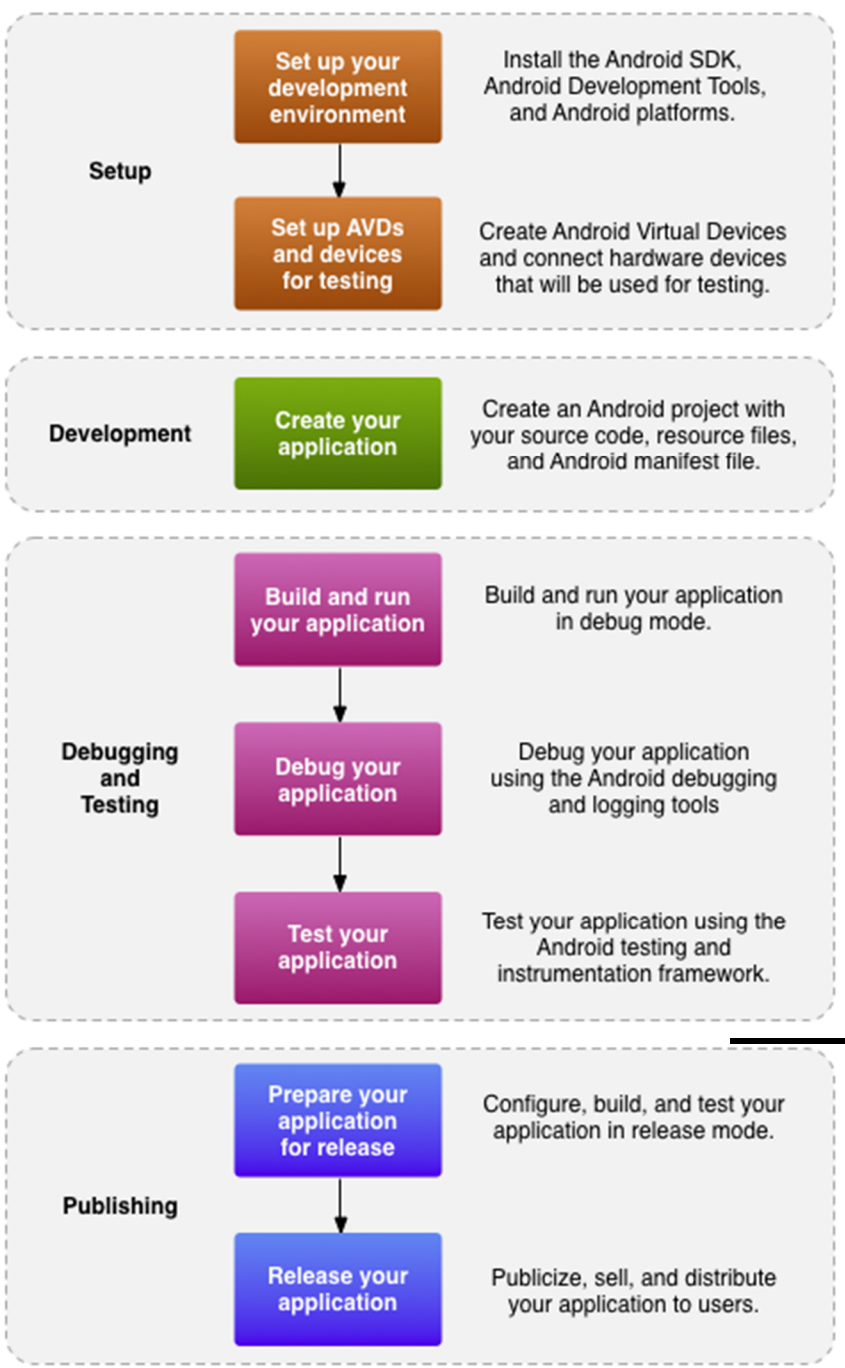


Android Studio

- Android builds evolved, with Gradle
  - Create multiple APKs for your Android app with different features using the same project.
  - Manage app dependencies with Maven.
  - Build APKs from Android Studio or the command line.



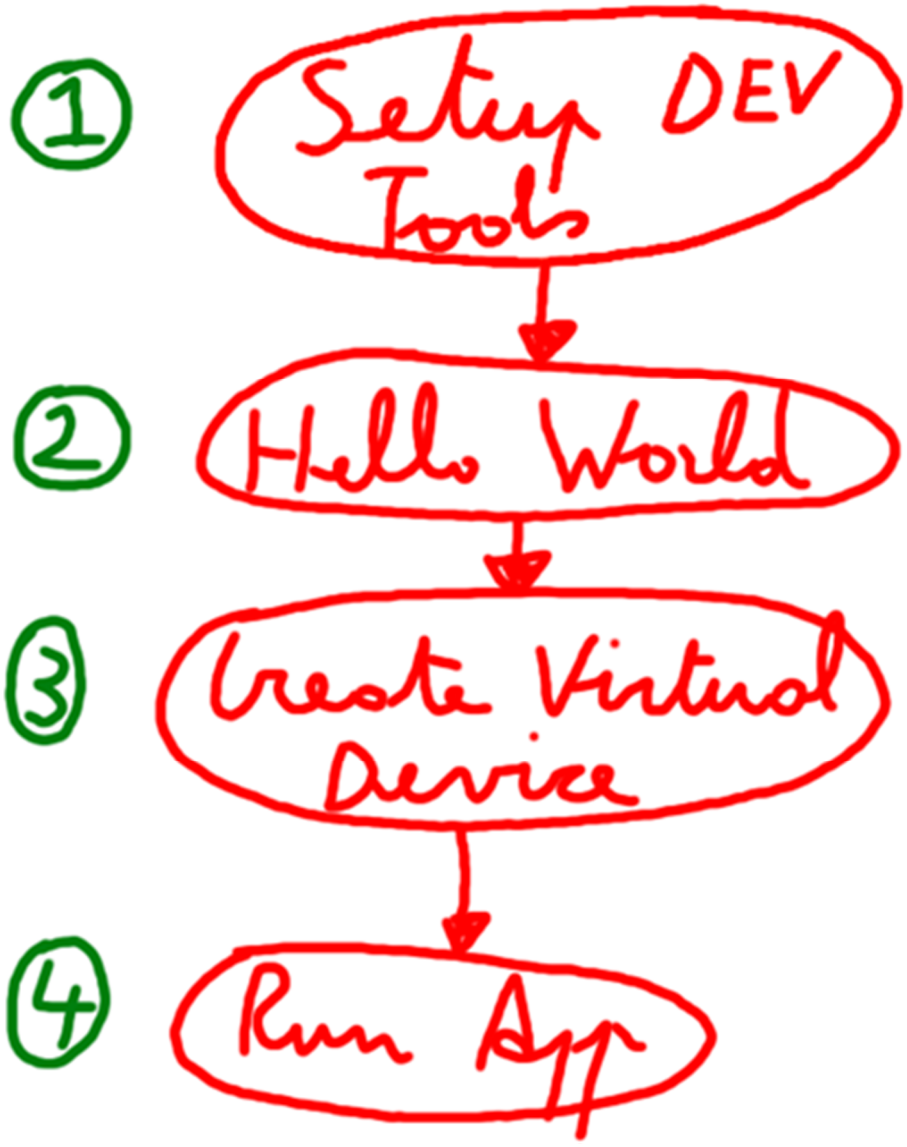
Development Workflow



We are going to do all of this

From this point on you're on your own

First Time Use Workflow



Android Development
Chapter 1 – Getting Started with Android Programming

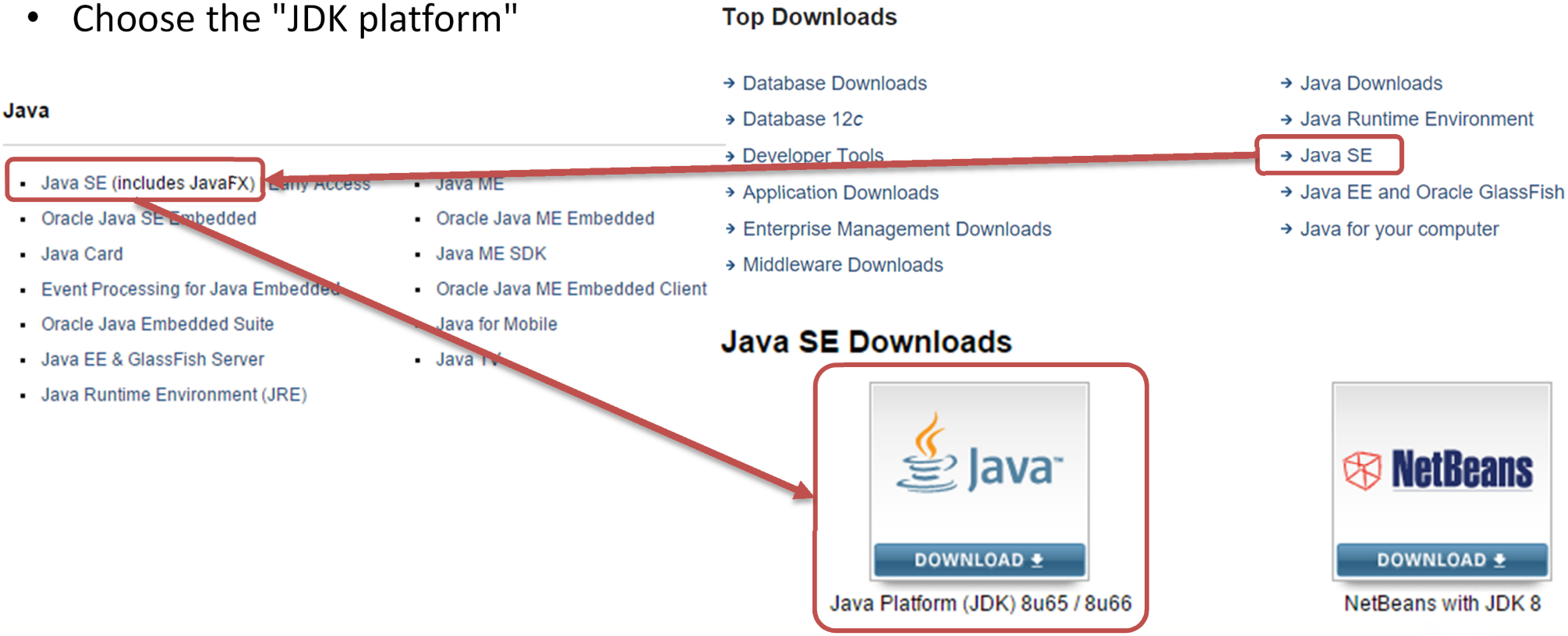
Step 1 - Setting Up Our Development Environment

katholieke hogeschool
associatie KU Leuven
vives

Step 1 - Setting Up Our Development Environment

Installing Java SE

- Start by installing the latest version of the Java Development Kit
  - Surf to https://www.oracle.com/downloads
  - Select "Java SE" from "Top Downloads"
  - Select "Java SE" from "Java" downloads
- Choose the "JDK platform"



Step 1 - Setting Up Our Development Environment

Installing Java SE

- Download the latest stable release (x32 or x64) (don't forget to accept the license agreement)

Java SE Development Kit 8u65

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM v6/v7 Hard Float ABI	77.69 MB	jdk-8u65-linux-arm32-vfp-hflt.tar.gz
Linux ARM v8 Hard Float ABI	74.66 MB	jdk-8u65-linux-arm64-vfp-hflt.tar.gz
Linux x86	154.67 MB	jdk-8u65-linux-i586.rpm
Linux x86	174.84 MB	jdk-8u65-linux-i586.tar.gz
Linux x64	152.69 MB	jdk-8u65-linux-x64.rpm
Linux x64	172.86 MB	jdk-8u65-linux-x64.tar.gz
Mac OS X x64	227.14 MB	jdk-8u65-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.71 MB	jdk-8u65-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.01 MB	jdk-8u65-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	140.22 MB	jdk-8u65-solaris-x64.tar.Z
Solaris x64	96.74 MB	jdk-8u65-solaris-x64.tar.gz
Windows x86	181.24 MB	jdk-8u65-windows-i586.exe
Windows x64	186.57 MB	jdk-8u65-windows-x64.exe

katholieke hogeschool associatie KU Leuven vives

Step 1 - Setting Up Our Development Environment

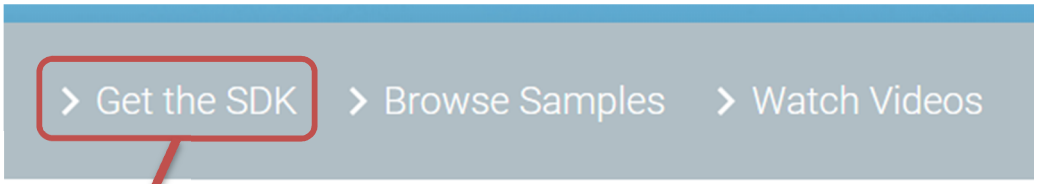
Installing Java SE

- Install the Java JDK before installing Android Studio.
- Restart your computer (!MUST!)

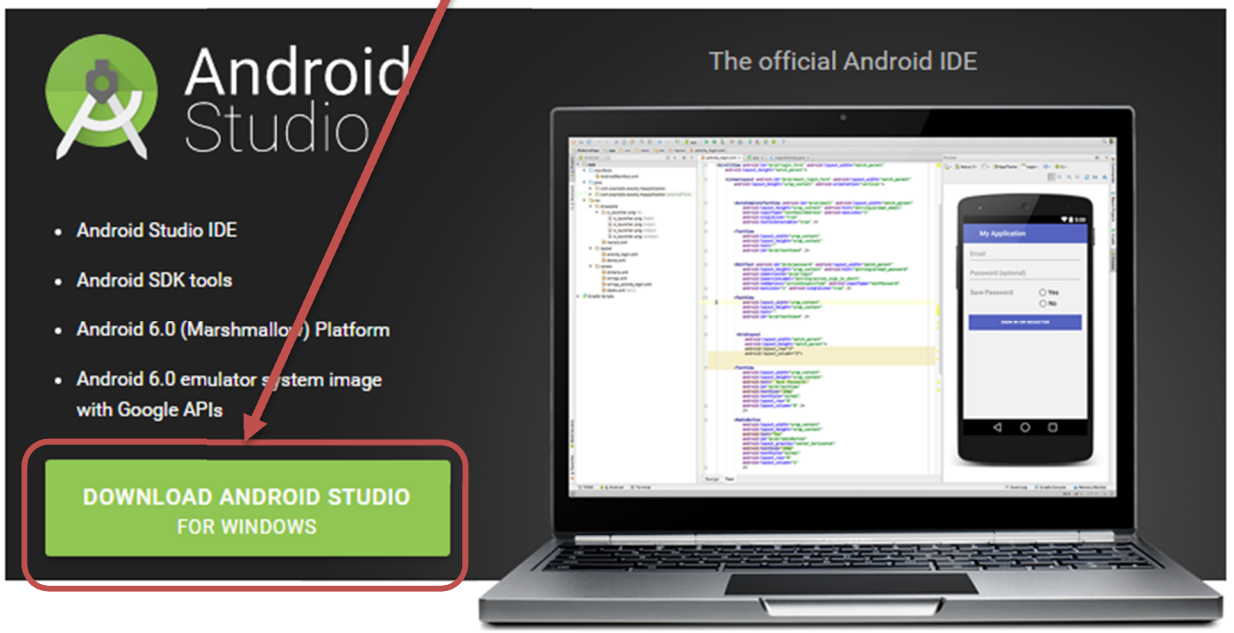
Step 1 - Setting Up Our Development Environment

Installing Android Studio

- To download Android Studio you need to surf to http://developer.android.com
- Click the "Get the SDK"



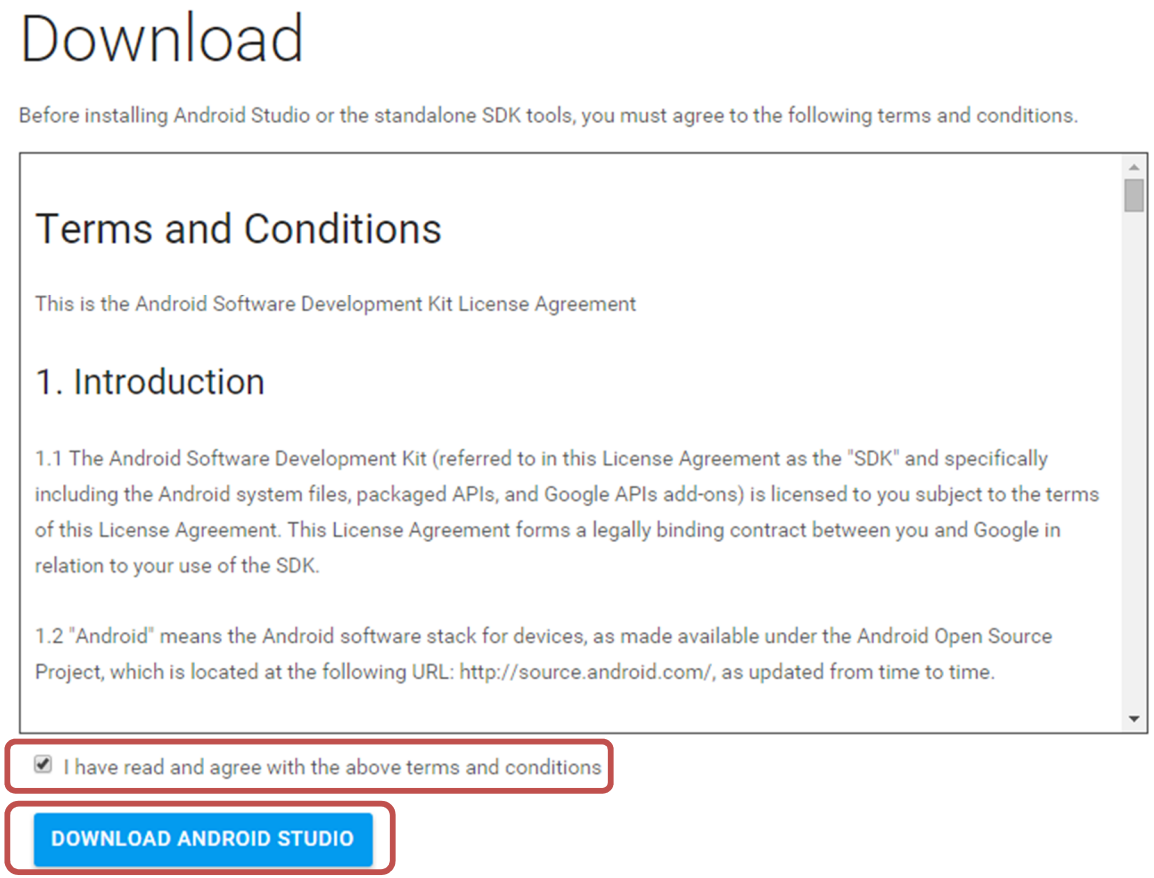
- Next choose "Download Android Studio"



Step 1 - Setting Up Our Development Environment

Installing Android Studio

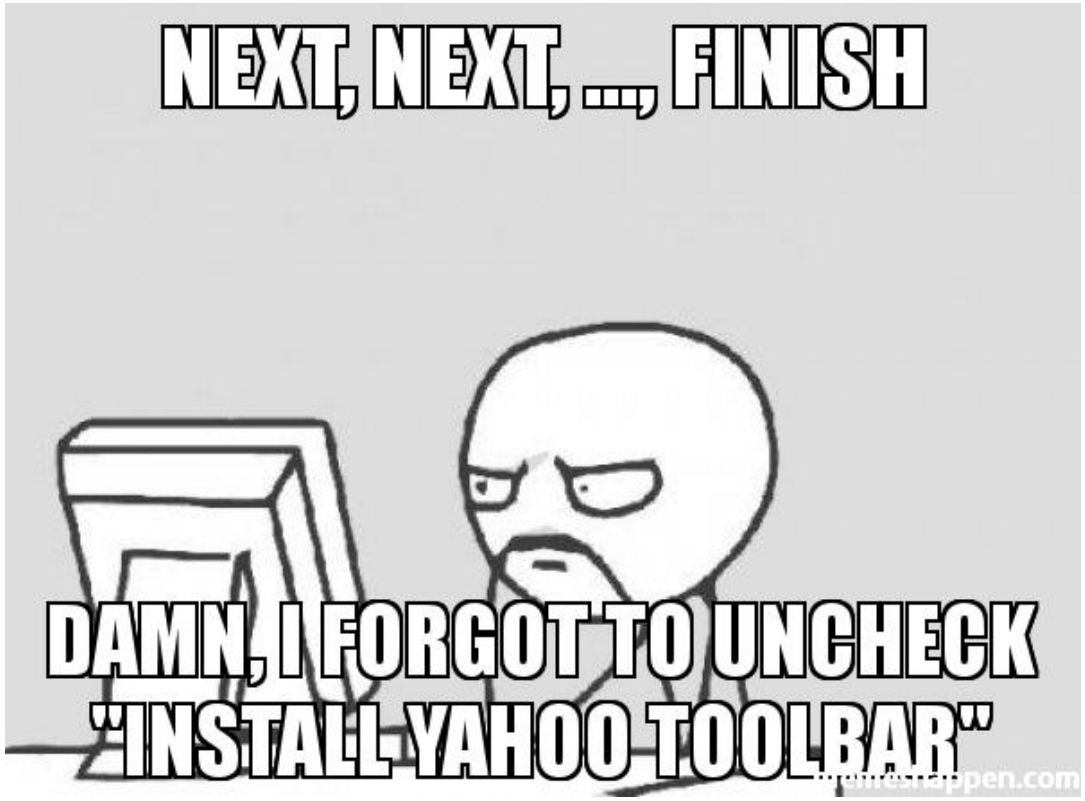
- Accept the license agreement and click "Download"



Step 1 - Setting Up Our Development Environment

Installing Android Studio

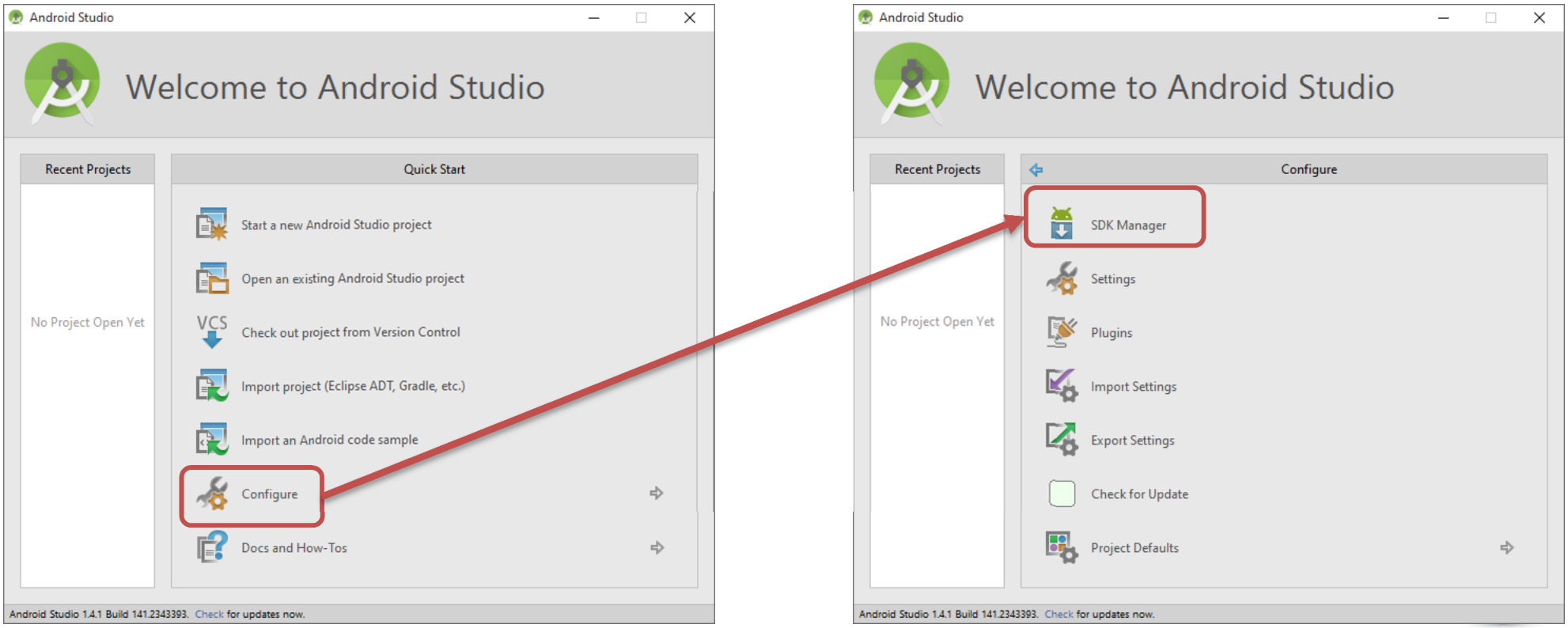
- Install Android Studio (Next -> Next -> .... -> Finish)



Step 1 - Setting Up Our Development Environment

Installing Android Development Tools

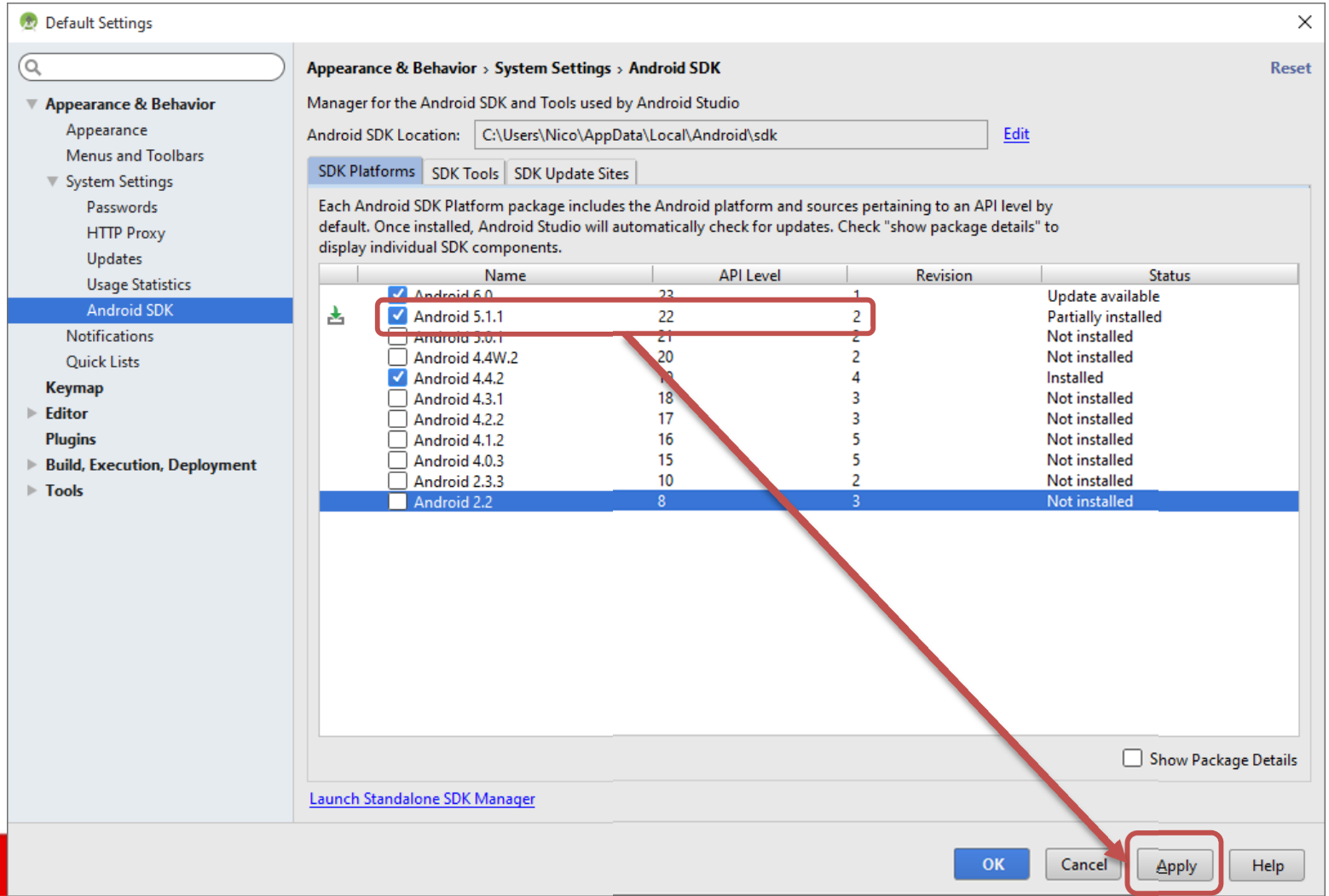
- Launch Android Studio
  - Choose "Configure"
  - Next pick "SDK Manager" to install Android SDK's



Step 1 - Setting Up Our Development Environment

Installing Android Development Tools

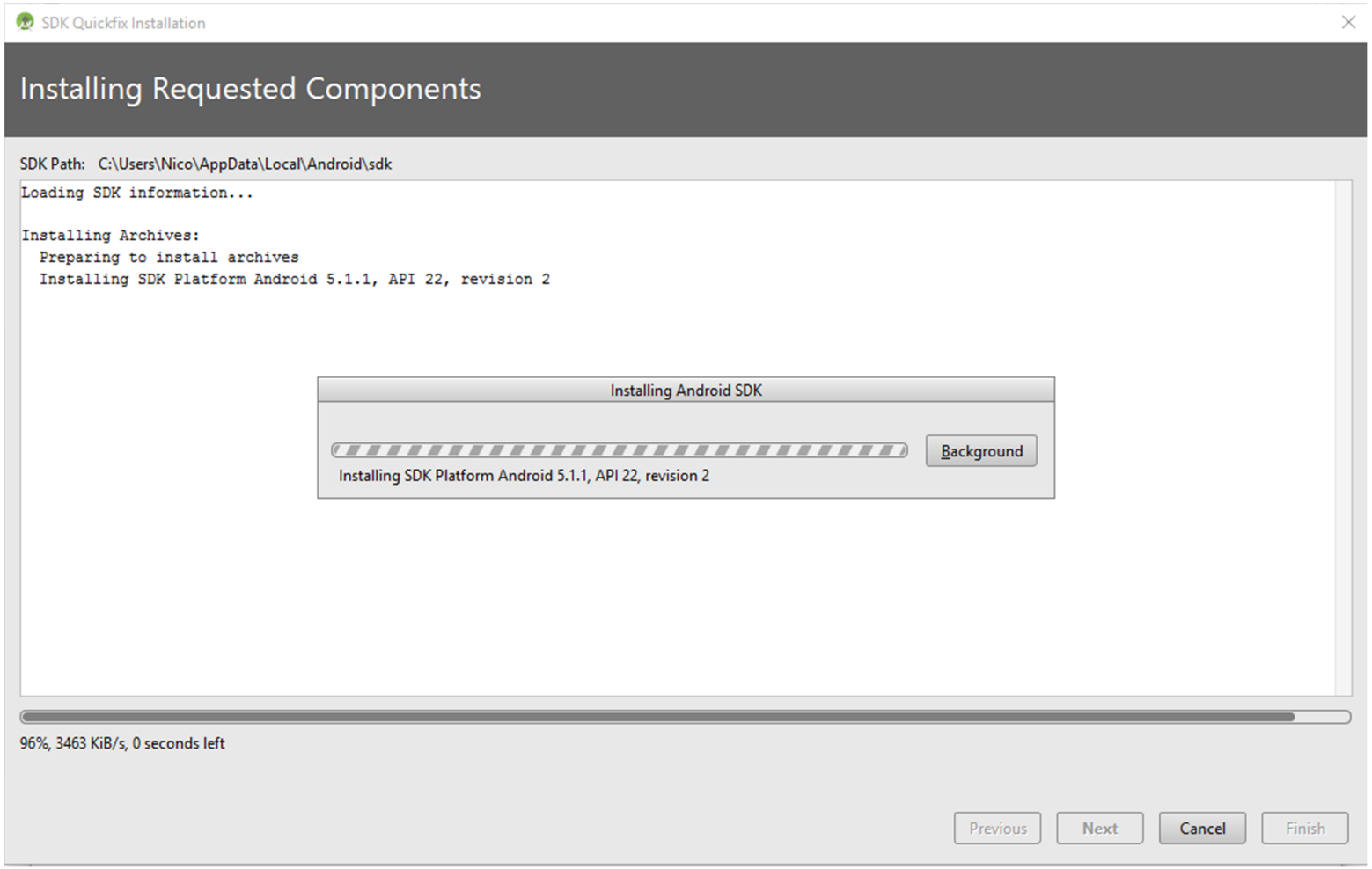
- Install Android 5.1.1 (API level 22)
  - Select "Android 5.1.1"
  - Hit "Apply"



Step 1 - Setting Up Our Development Environment

Installing Android Development Tools

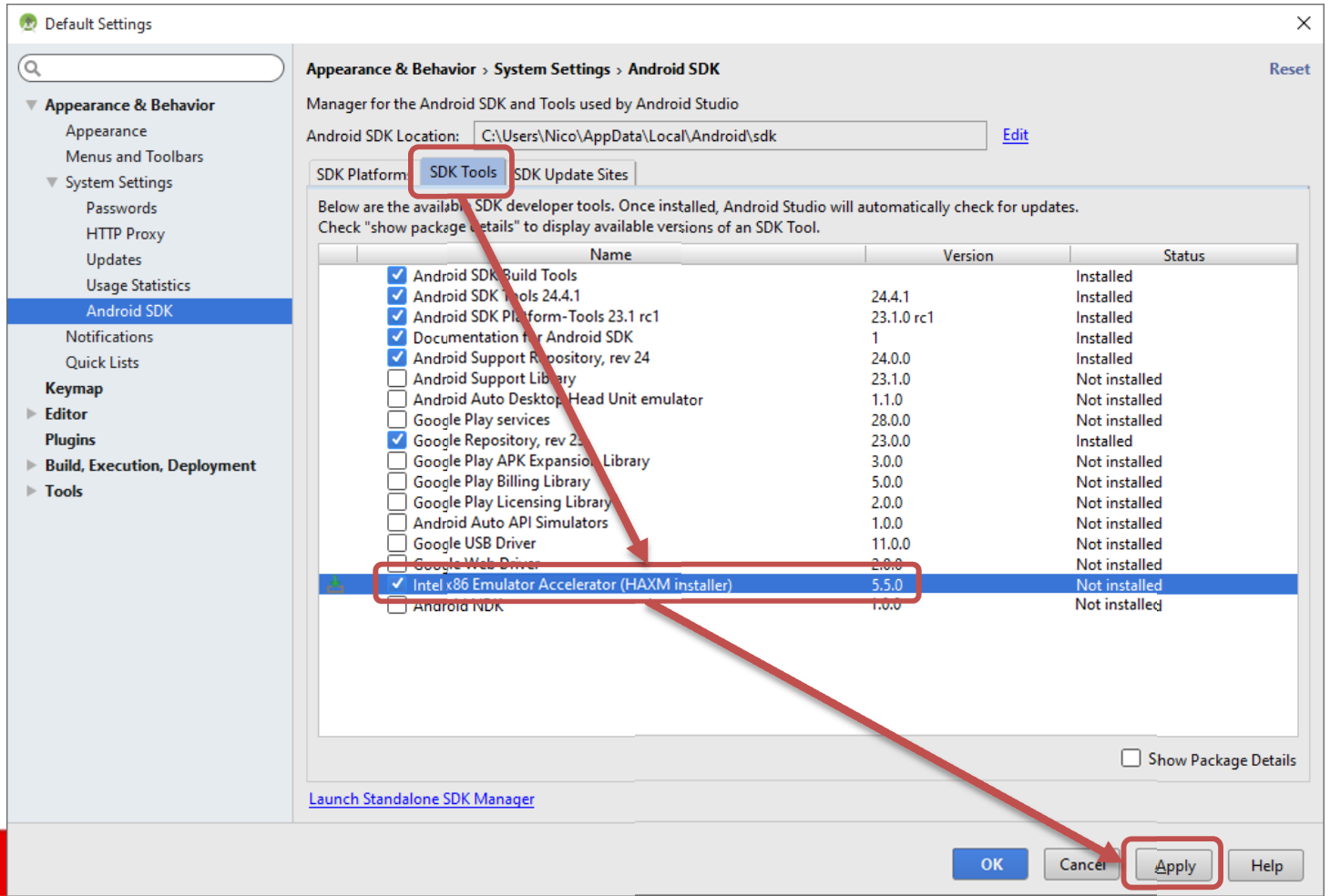
- Wait for the download and installer to finish



Step 1 - Setting Up Our Development Environment

Installing Intel x86 Emulator Accelerator (HAXM Installer)

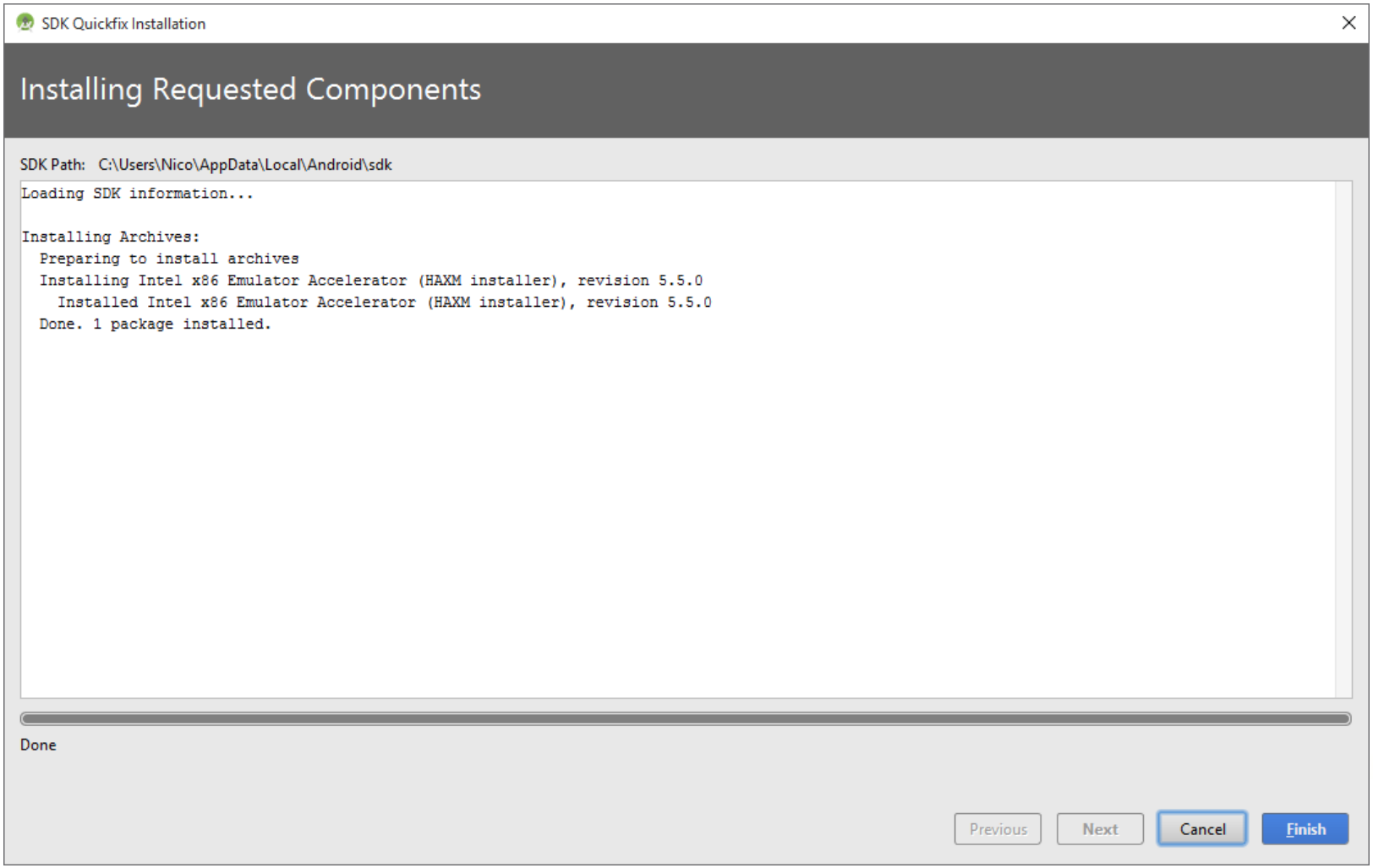
- Install Emulator Accelerator
  - Select the SDK Tools Tab
  - Check "Installing Intel x86 Emulator Accelerator (HAXM Installer)"
  - Select "Apply"



Step 1 - Setting Up Our Development Environment

Installing Intel x86 Emulator Accelerator (HAXM Installer)

- Wait for the download and installer to finish



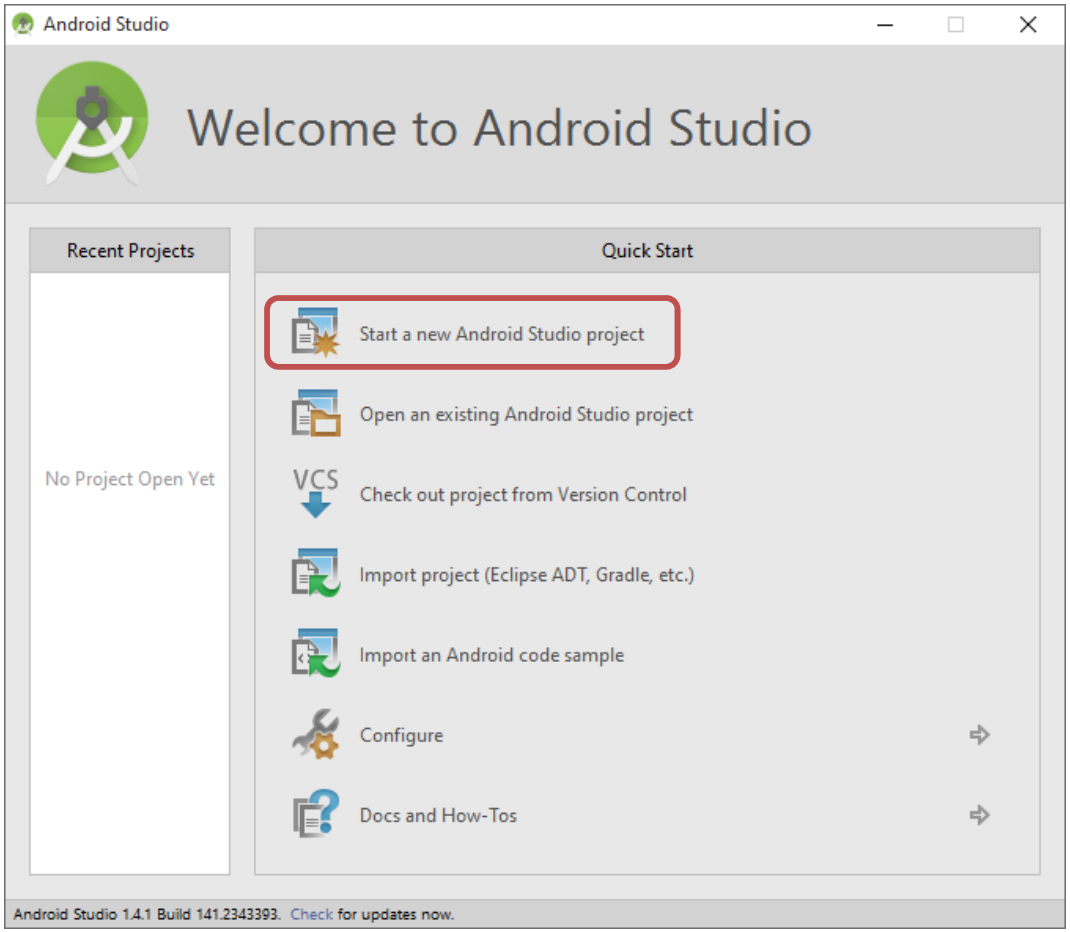
Android Development
Chapter 1 – Getting Started with Android Programming

Step 2 - Create a Hello World Application

katholieke hogeschool
associatie KU Leuven
vives

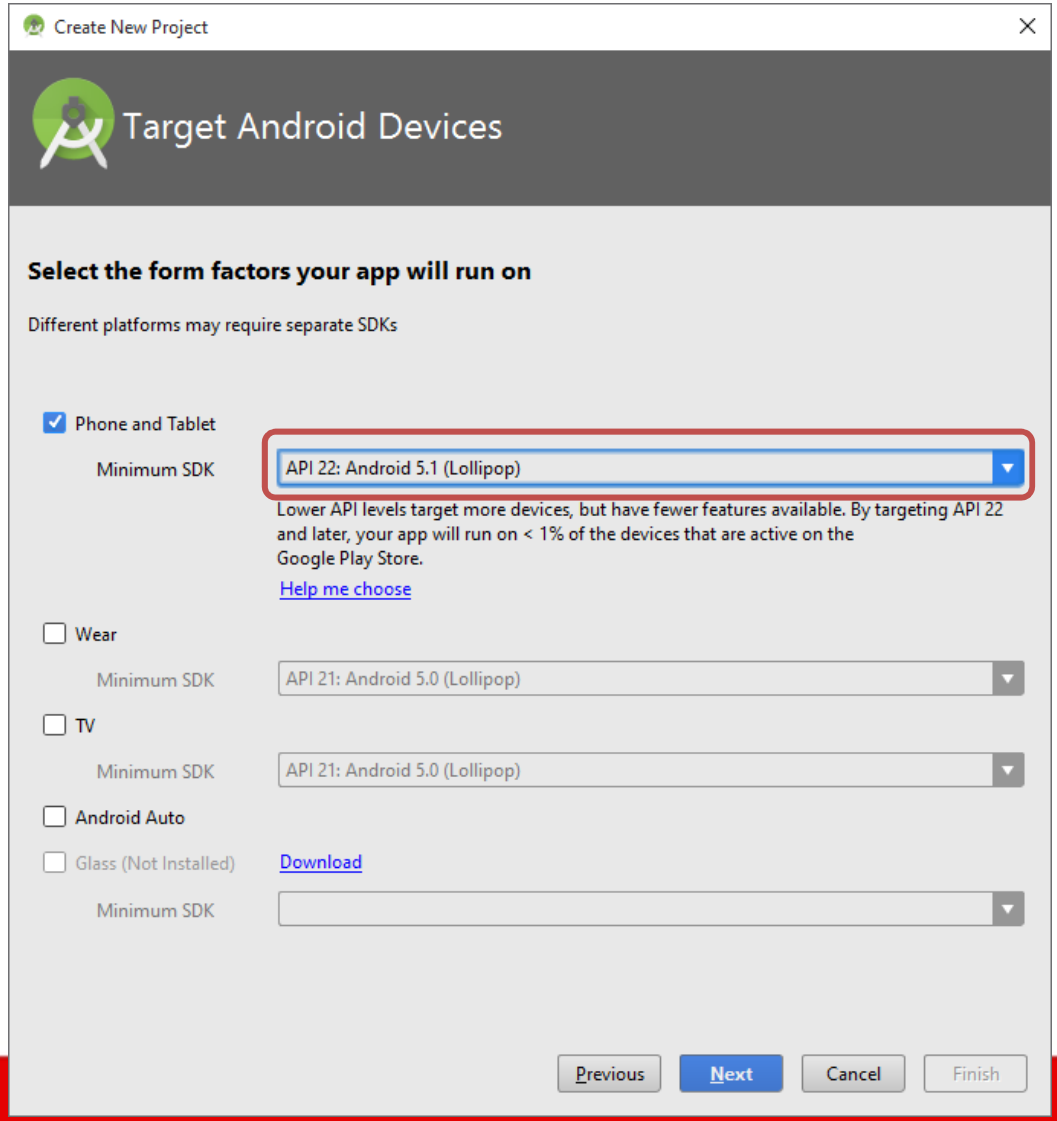
Step 2 - Create a Hello World Application

- Start Android Studio
- Select "start a new Android Studio project"



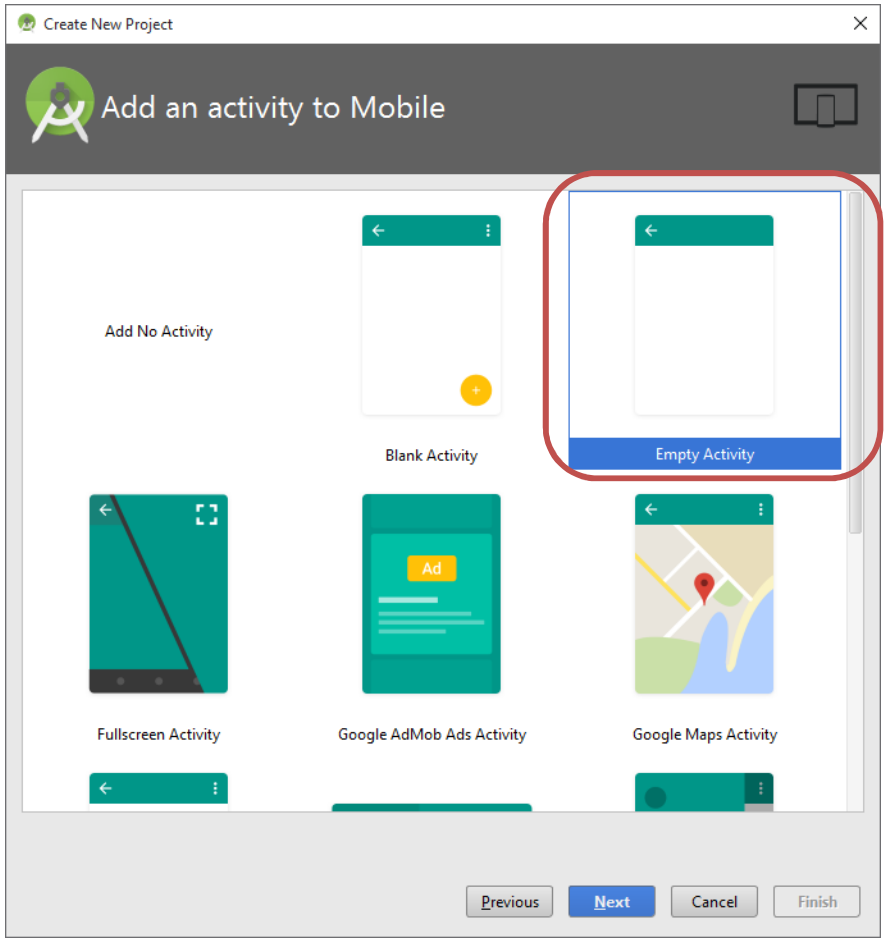
Step 2 - Create a Hello World Application

- Select API 22 (Android 5.1)



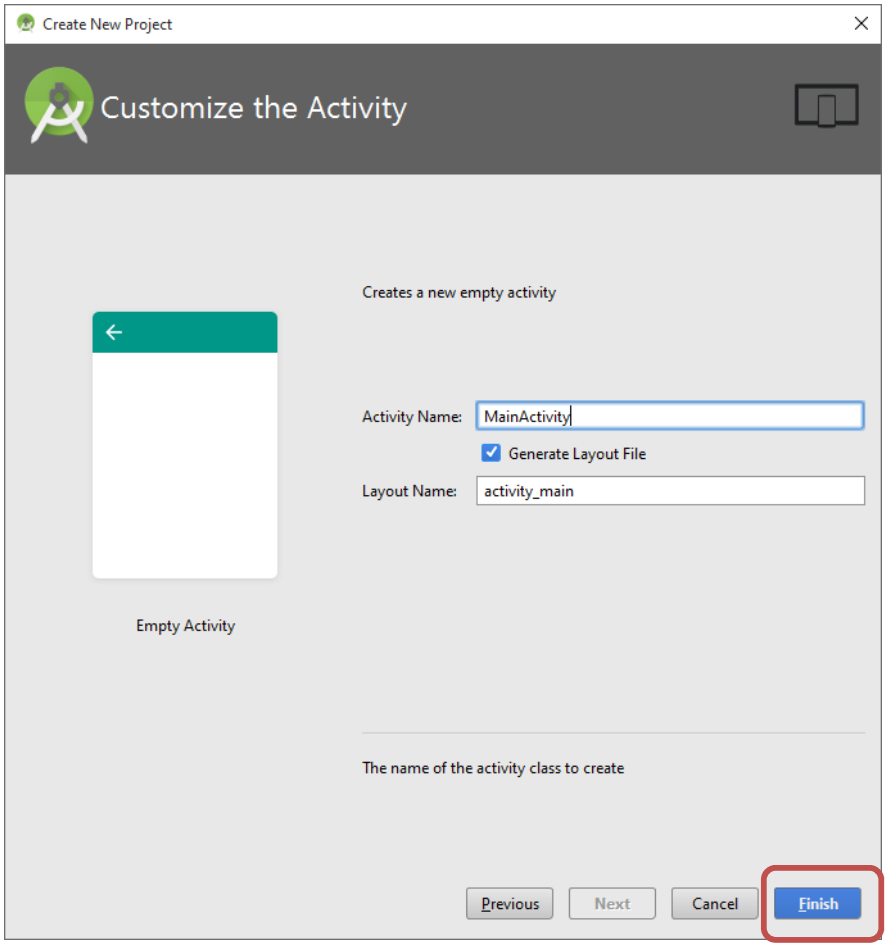
Step 2 - Create a Hello World Application

- Next make sure to select the "Empty Activity" template
- You can think of an activity as a graphical window shown to the user



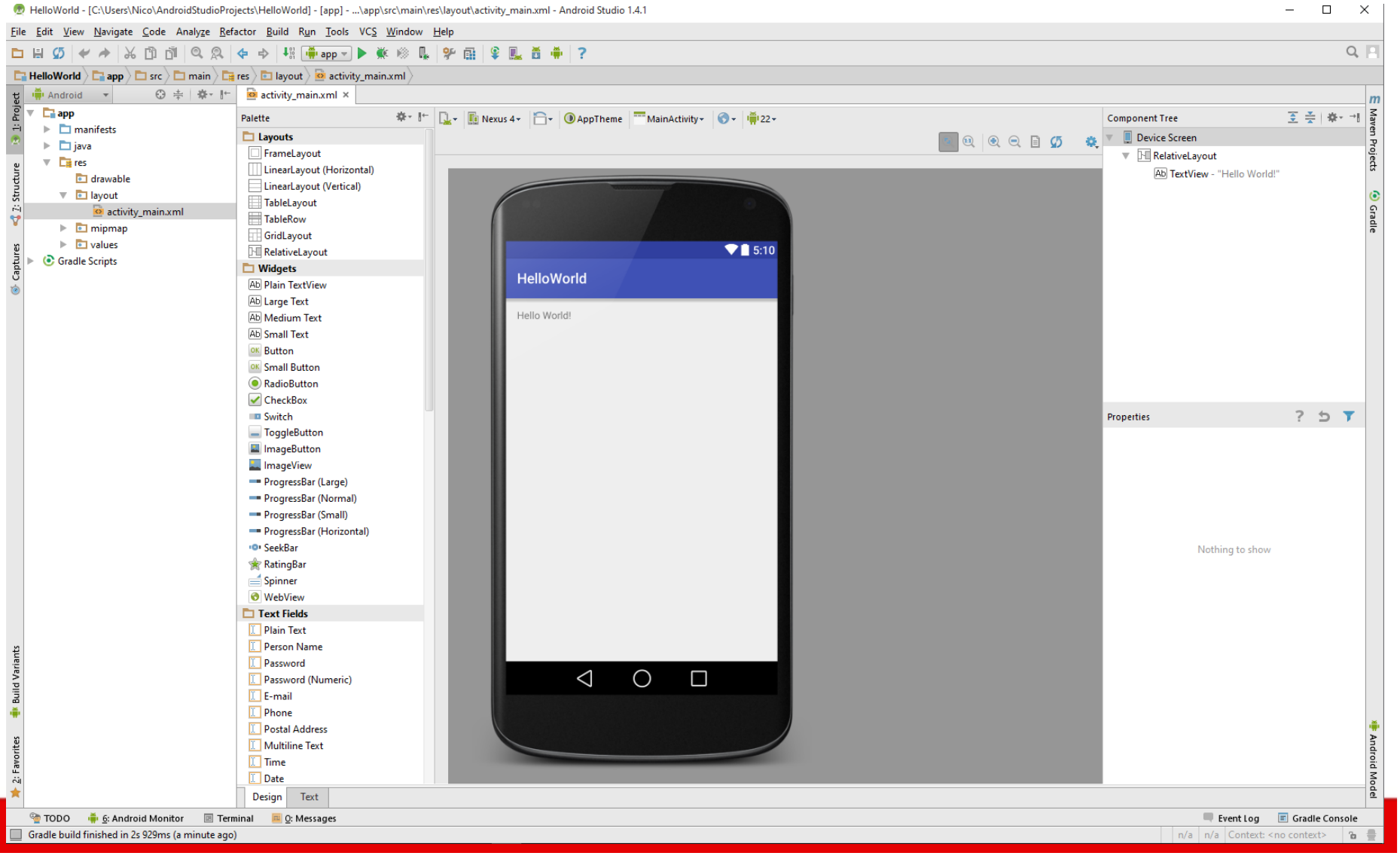
Step 2 - Create a Hello World Application

- Give the activity a name
  - Leave the name for the main activity as is



Step 2 - Create a Hello World Application

- Your project should open now



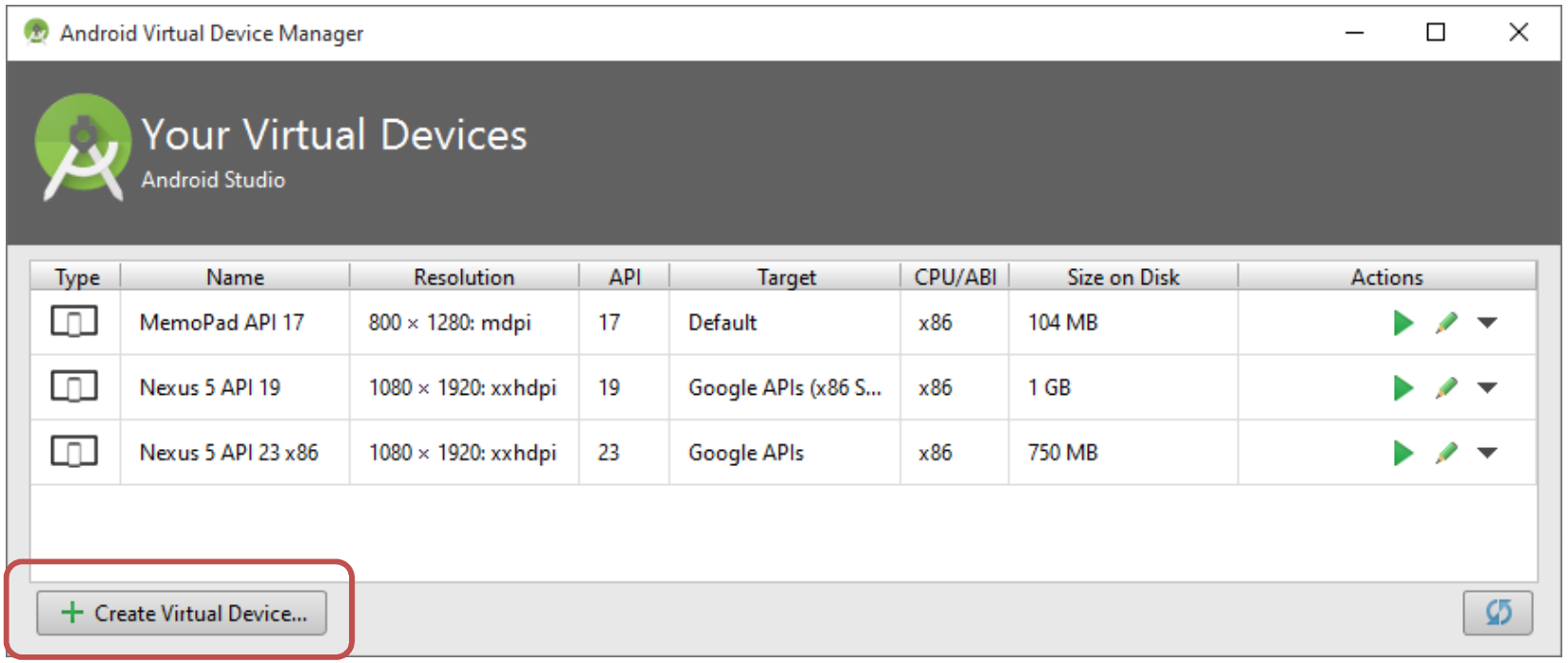
Android Development
Chapter 1 – Getting Started with Android Programming

Step 3 - Creating an Android Virtual Device

katholieke hogeschool
associatie KU Leuven
vives

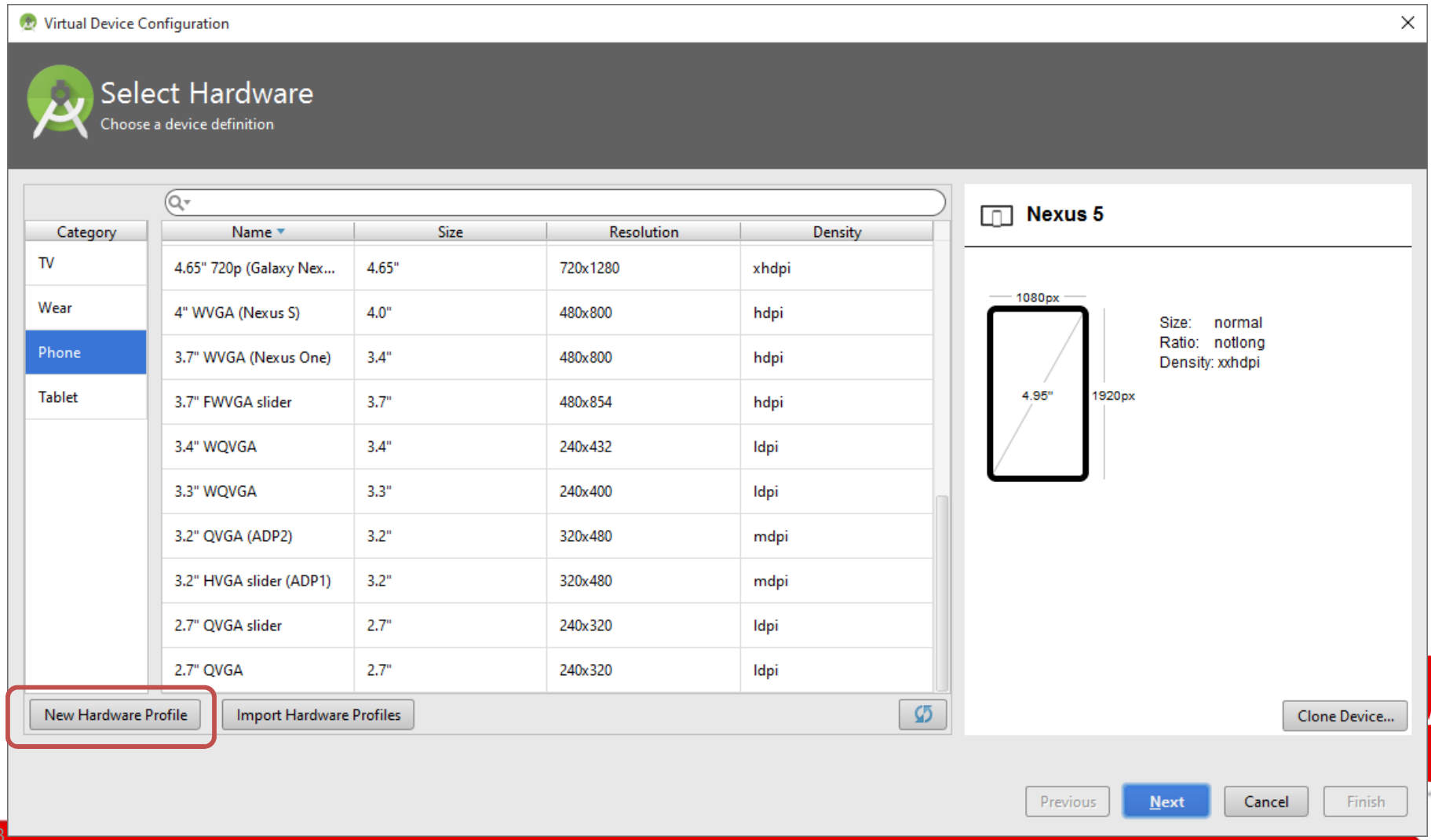
Step 3 - Creating an Android Virtual Device

- Android applications may be run on a real device or on the Android Emulator, which ships with the Android SDK.
- Let's create an Android Virtual Device (AVD)
- Go to "Tools" => "Android" => "AVD Manager"
- Click "Create Virtual Device ..."



Step 3 - Creating an Android Virtual Device

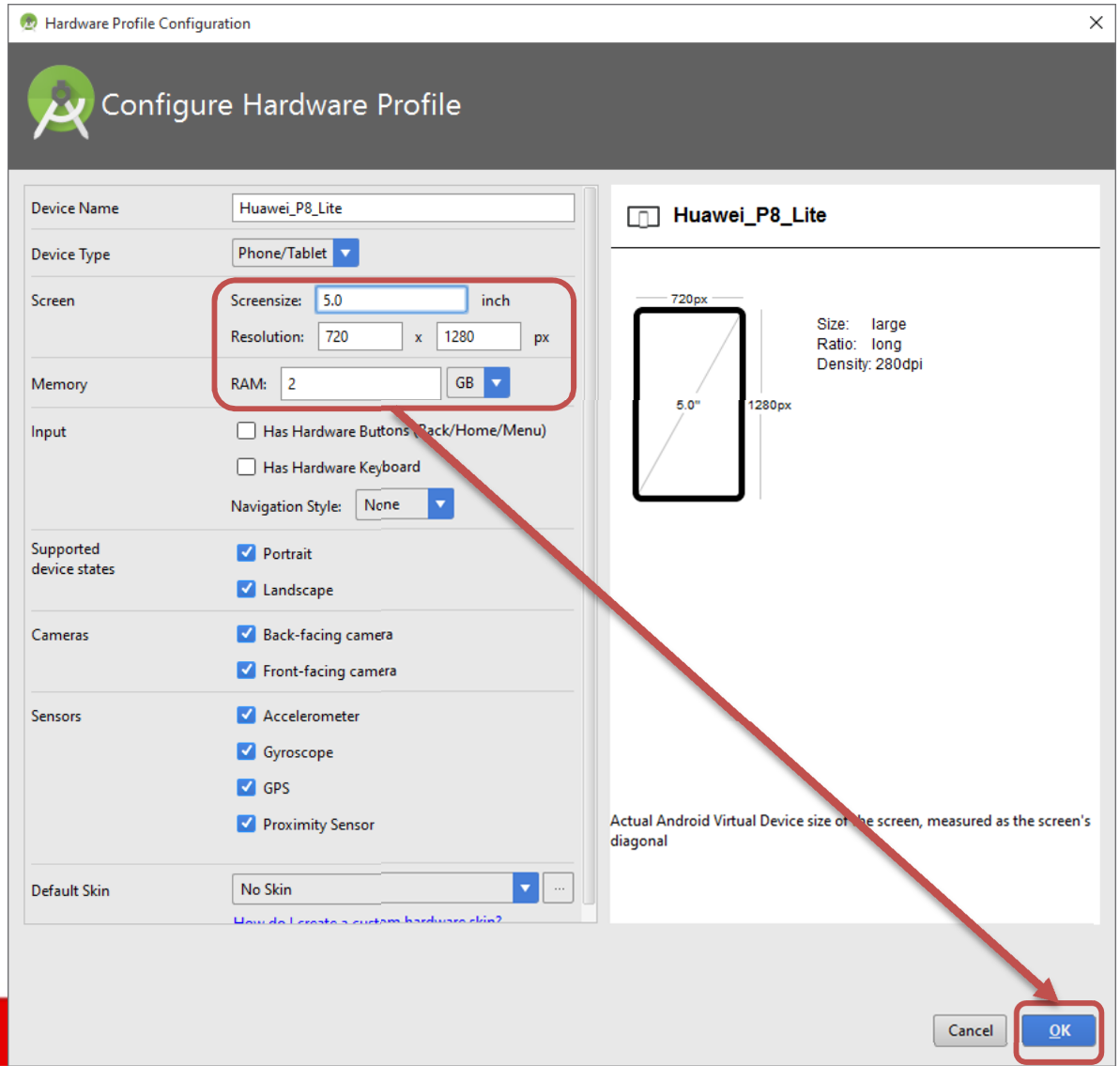
- Click "New Hardware Profile"



Step 3 - Creating an Android Virtual Device

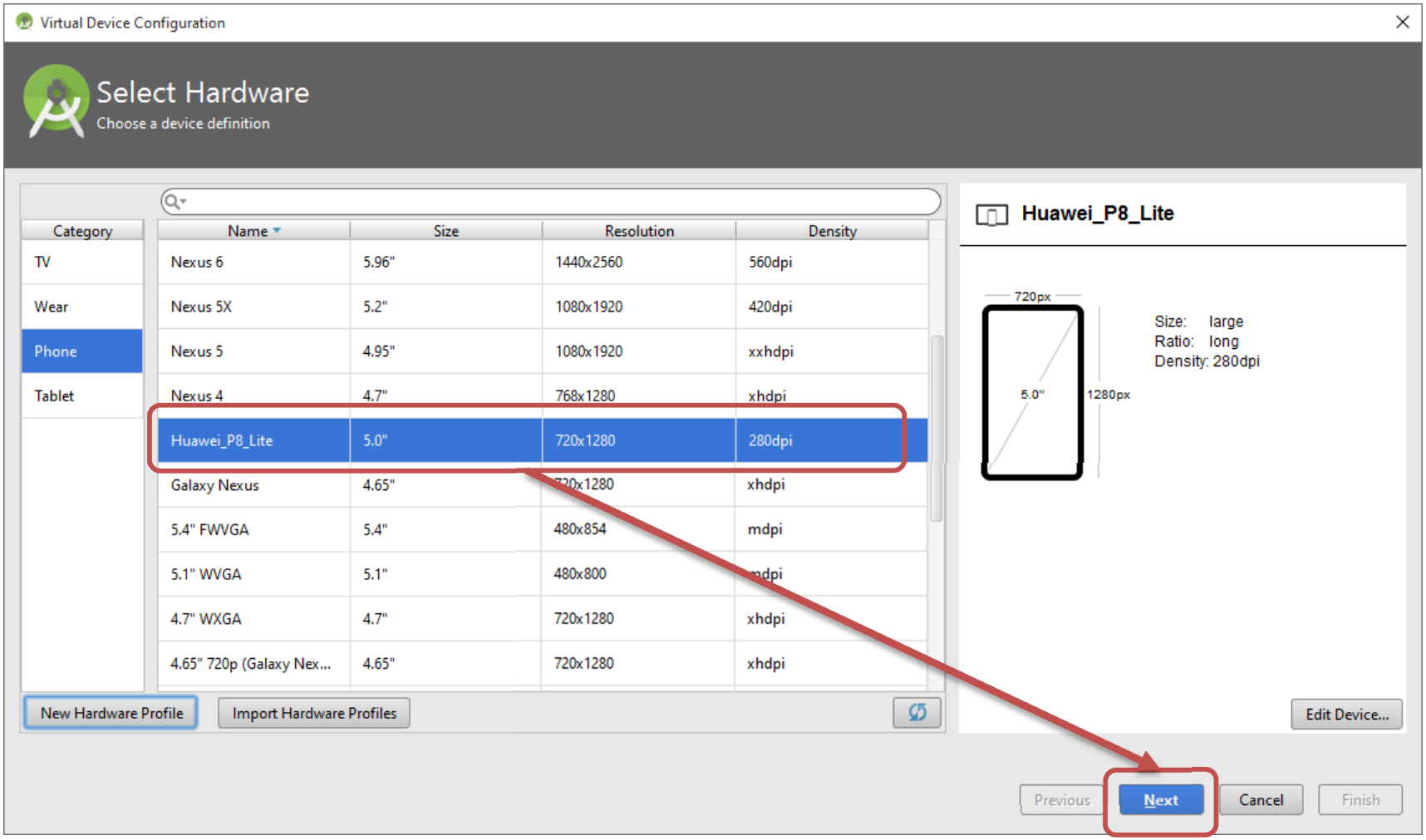
- Let's create a profile for the Huawei P8 Lite
  - Screensize = 5 inch
  - Resolution = 720 x 1280
  - RAM = 2GB

- Click "OK"



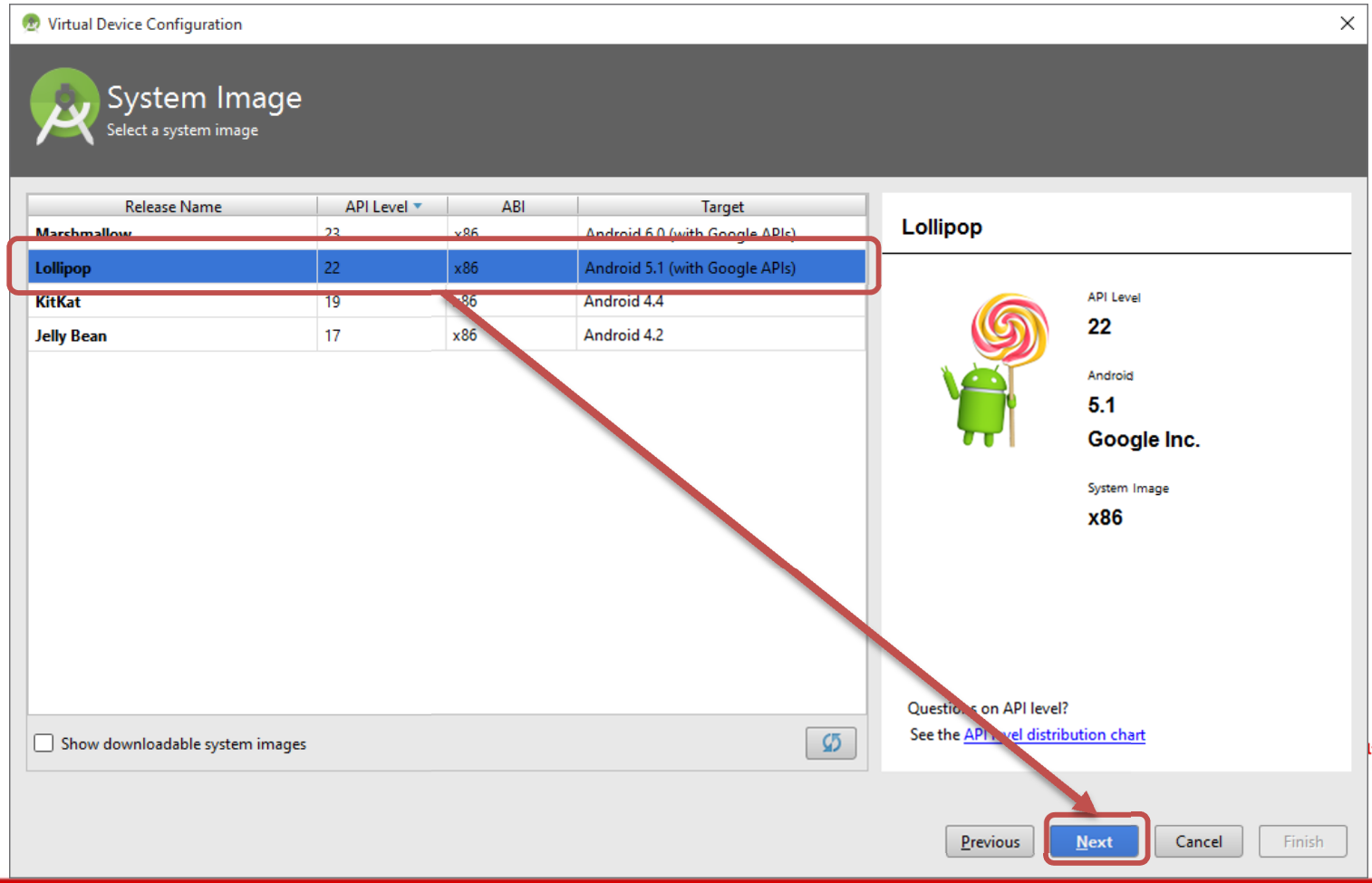
Step 3 - Creating an Android Virtual Device

- Select your new hardware profile and click "Next"



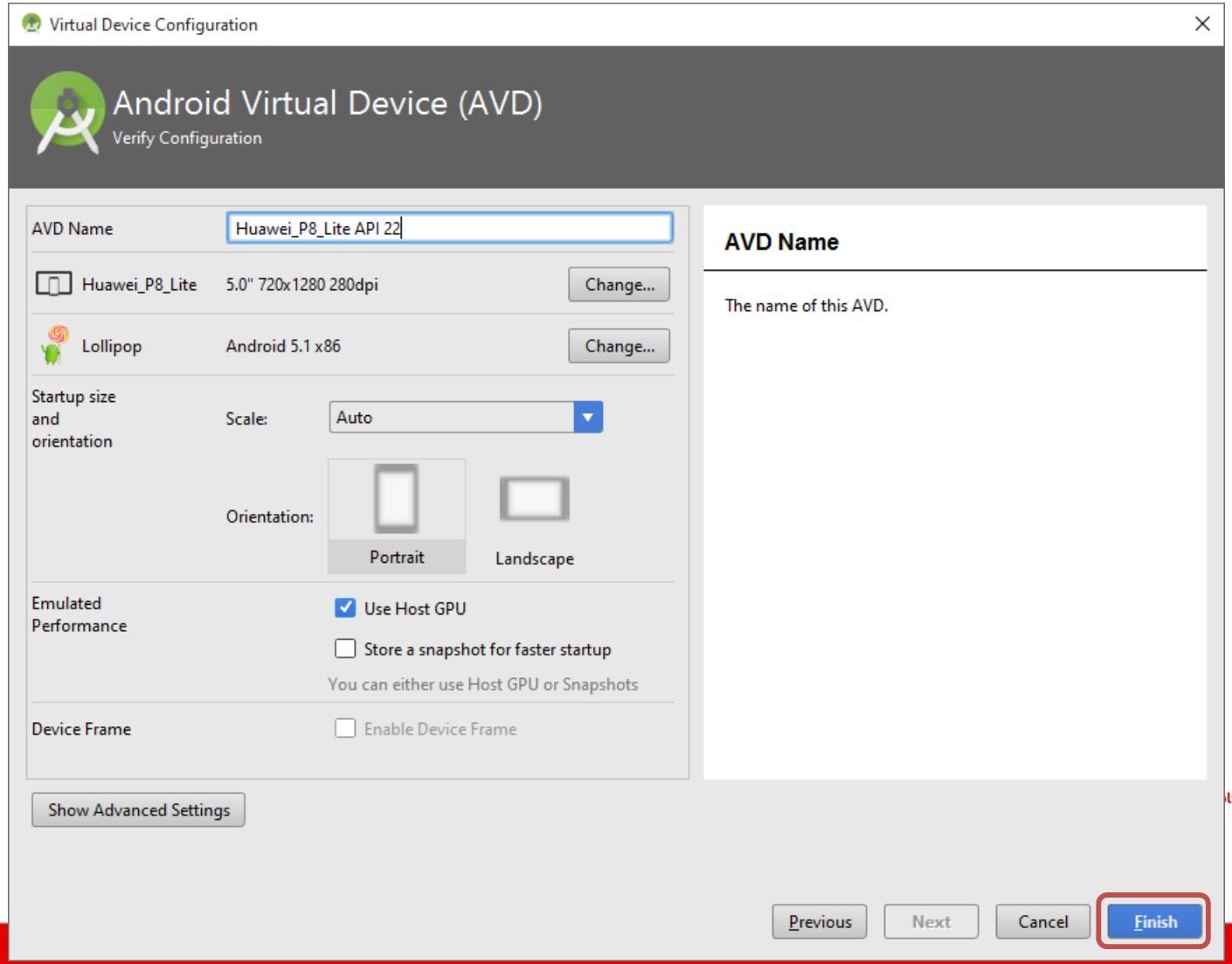
Step 3 - Creating an Android Virtual Device

- Select the Android version you want to run on your AVD and hit "Next"
  - In our case Android 5.1 (API 22)



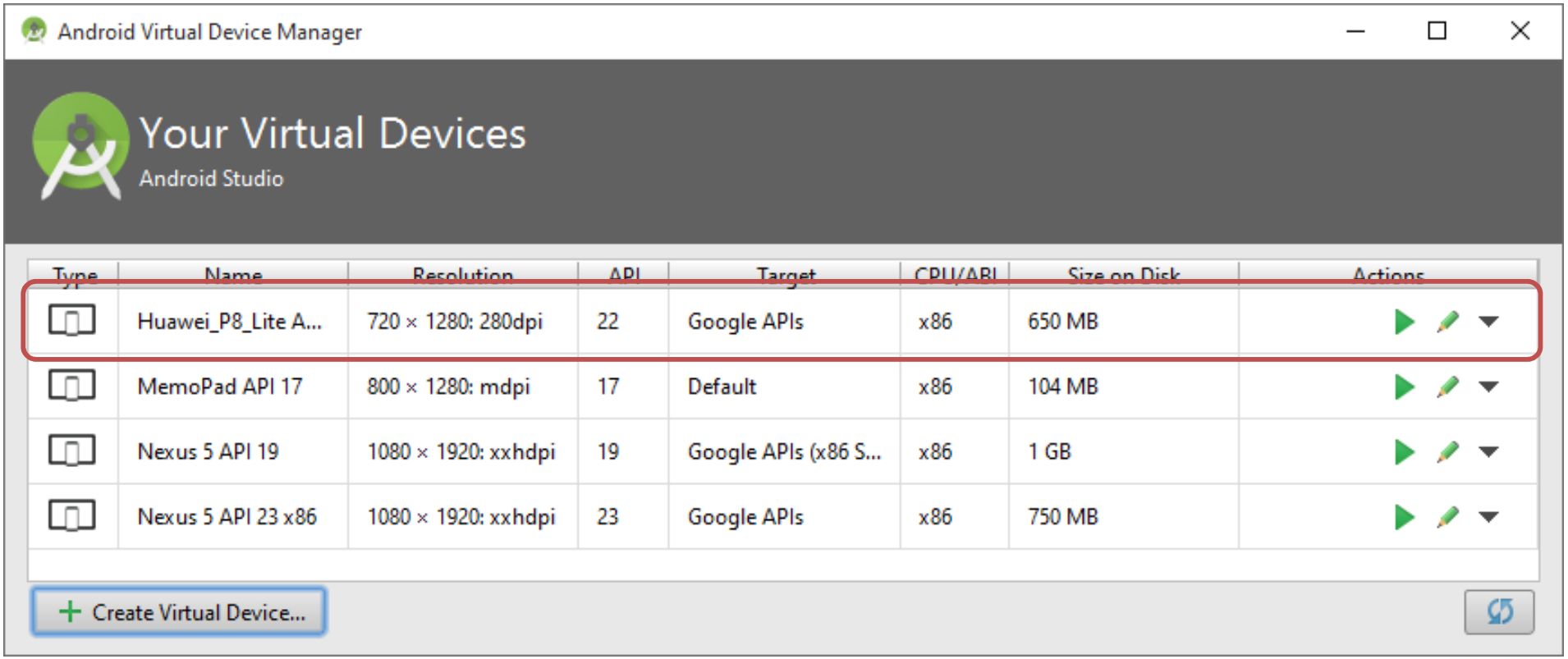
Step 3 - Creating an Android Virtual Device

- Leave the default settings and hit "Finish"



Step 3 - Creating an Android Virtual Device

- Your new AVD should appear in the list
  - Close the AVD window



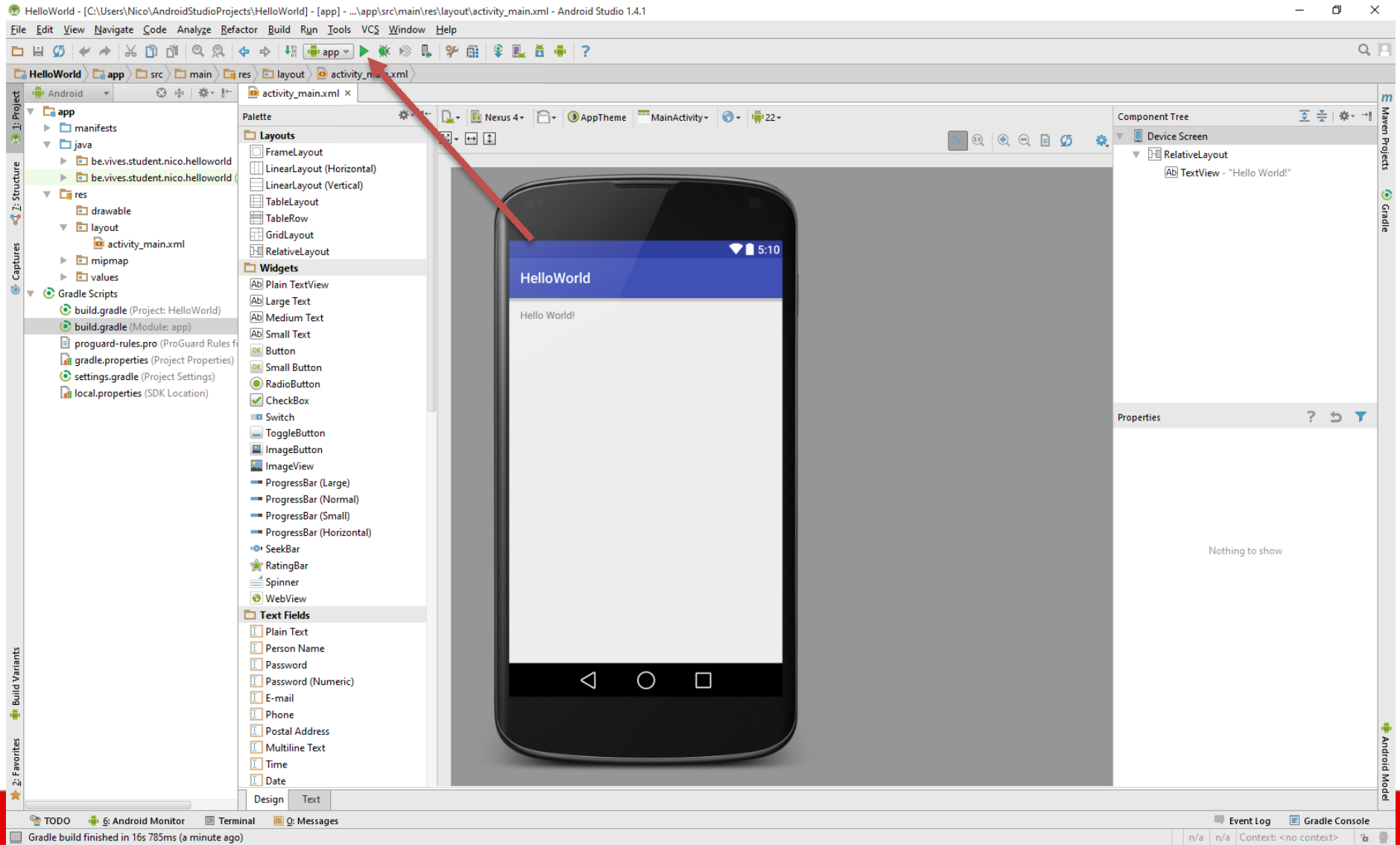
Android Development
Chapter 1 – Getting Started with Android Programming

Step 4 - Run Your Application on the AVD

katholieke hogeschool
associatie KU Leuven
vives

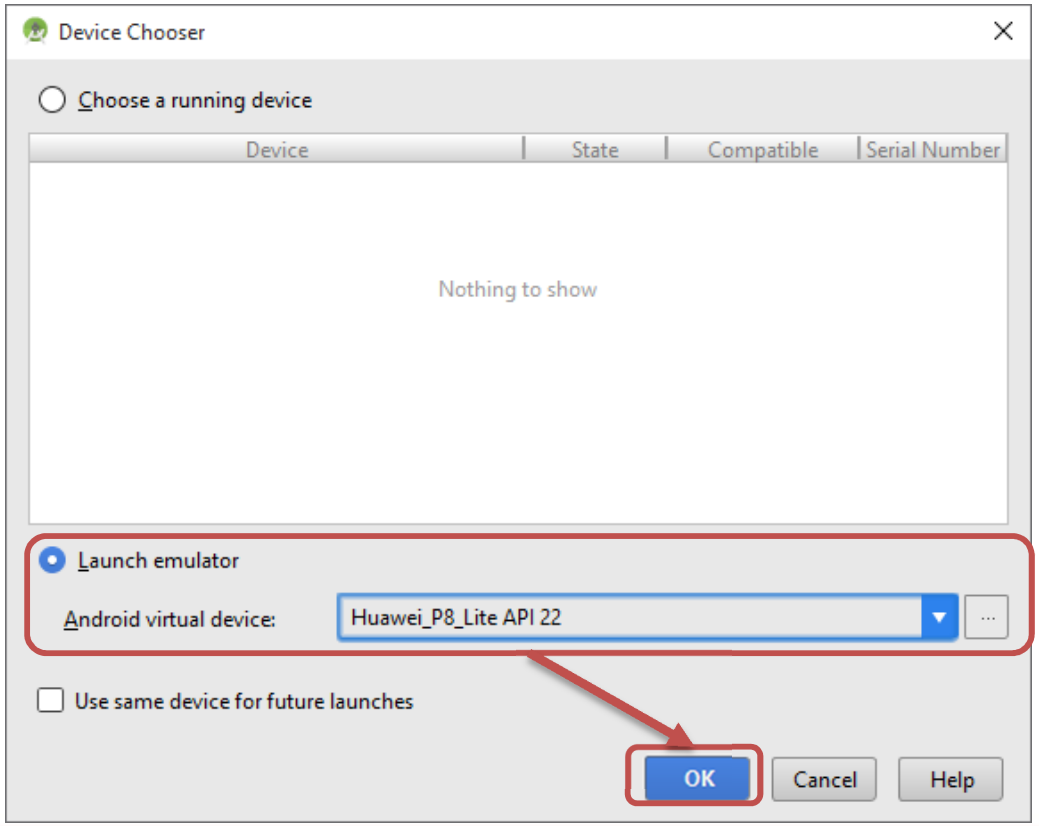
Step 4 - Run Your Application on the AVD

- Click the green play button to run your application



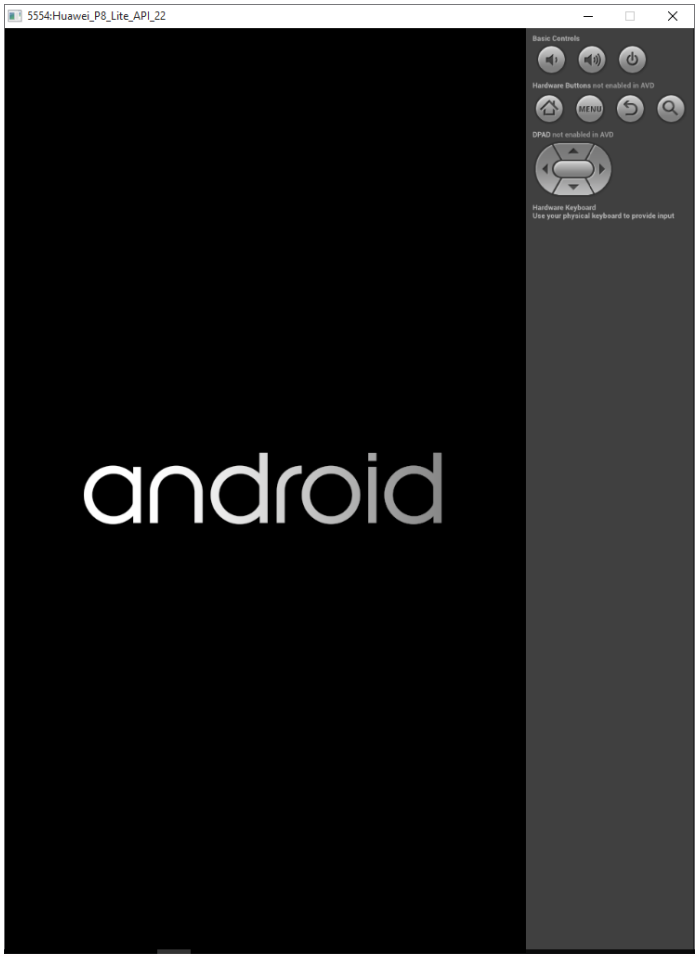
Step 4 - Run Your Application on the AVD

- The "Device Chooser" should appear
  - Select your custom virtual device from the dropdown and click "OK"
  - Next time you can choose the AVD from the running devices if you don't close the AVD between tests



Step 4 - Run Your Application on the AVD

- The AVD should start and your application should show
  - You may need to swipe up to see your app



Android Development
Chapter 1 – Getting Started with Android Programming

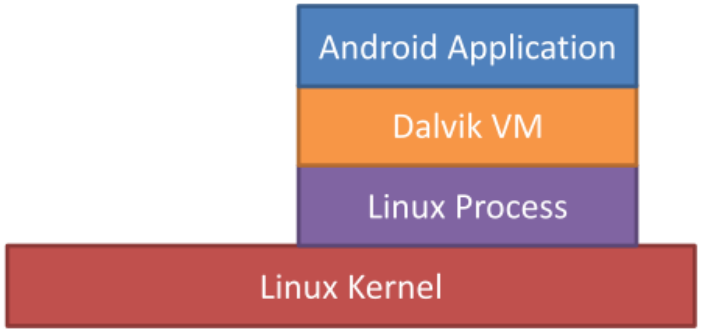
Anatomy of an Android Application

katholieke hogeschool
associatie KU Leuven
vives

Application Fundamentals

- Development Language
  - Java
  - Android SDK tools compile the code into an Android package, an archive file with an .apk suffix

- Security sandbox
  - Each application has a unique Linux user ID
  - Each process has its own virtual machine (VM)
  - Every application runs in its own Linux process

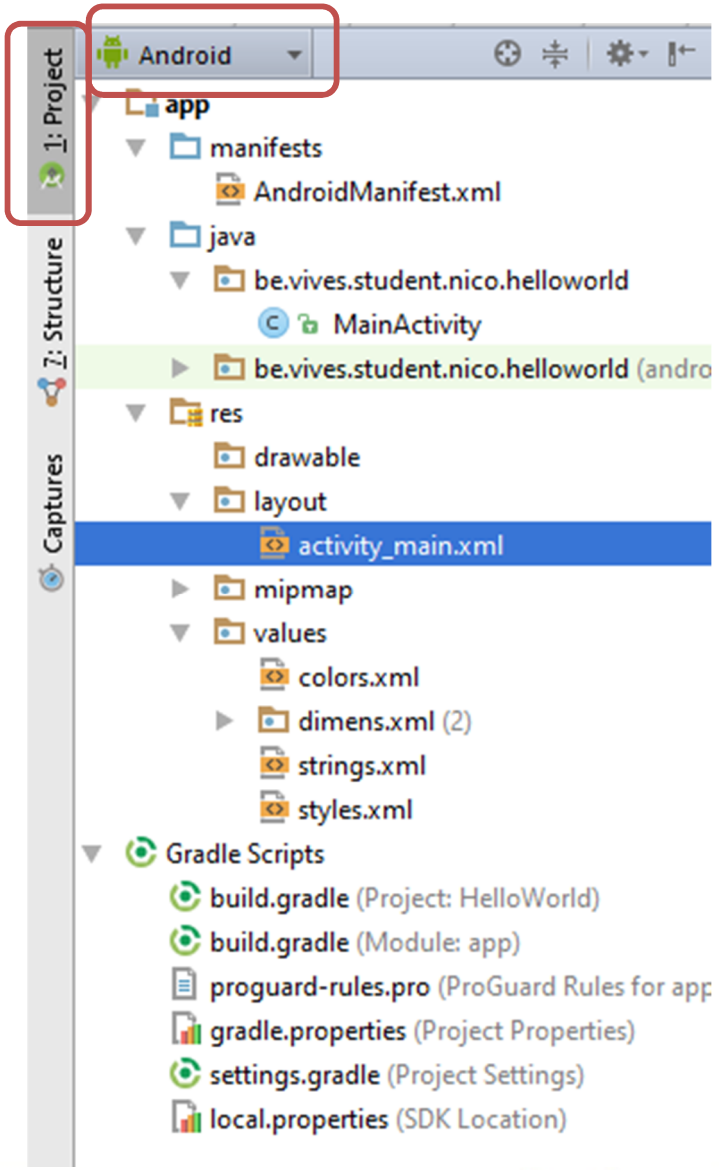


- Principle of least privilege
  - Each application, has access only to the components that it requires to do its work and no more.

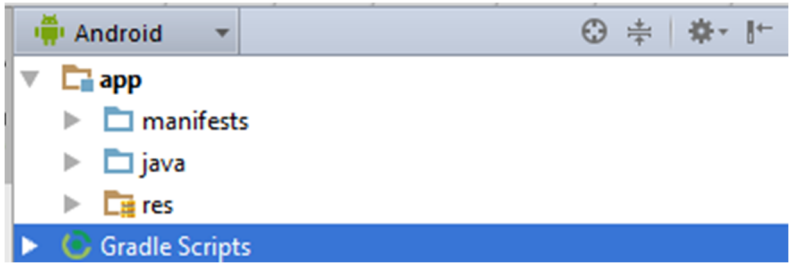
Anatomy of an Android Application

- Show project pane by selecting "Project" on left side
- Make sure to select the "Android" view

- The Android project view in Android Studio shows a flattened version of your project's structure that provides quick access to the key source files of Android projects
  - Do note that the project structure on disk differs from this representation.



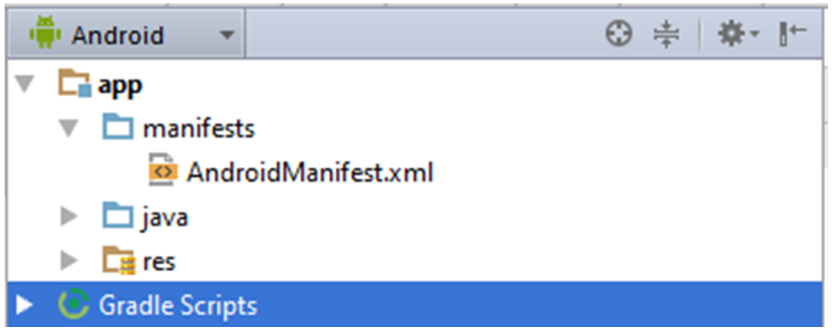
Anatomy of an Android Application



- At the top level we find the following elements
  - manifests
    - Manifest files
      - Control files that describes the nature of the application and each of its components
  - java
    - Contains the java source code files
  - res
    - Contains all the resource files used in the application such as images (multiple resolutions), GUI layout and string literals

Anatomy of an Android Application

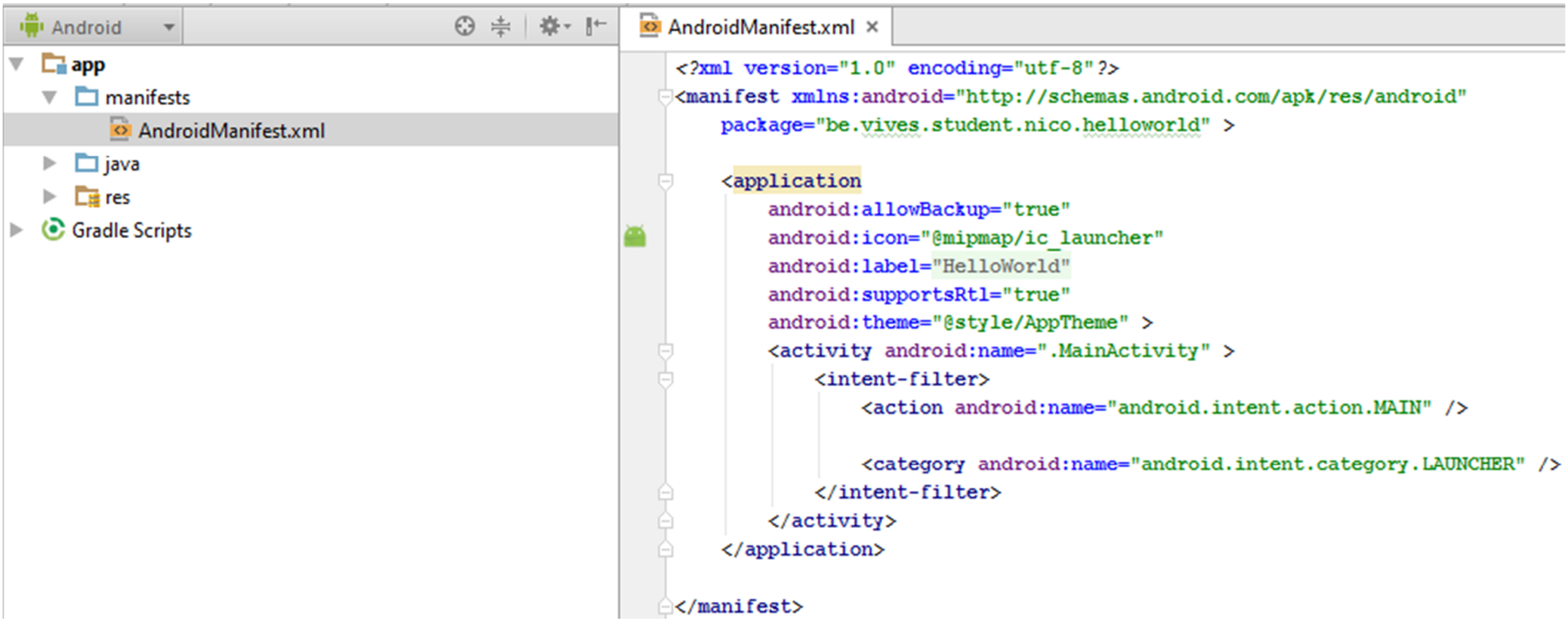
Manifests



- AndroidManifest.xml
  - This is the control file that describes the nature of the application and each of its components.
  - It describes
    - certain qualities about the activities, services, intent receivers, and content providers;
    - what permissions are requested;
    - what external libraries are needed;
    - what device features are required,
    - what API Levels are supported or required;
    - The entry point of the application, defined inside an intent-filter. This defines how to application can be started and what the entry point is.
- Cfr.: http://developer.android.com/guide/topics/manifest/manifest-intro.html

Anatomy of an Android Application

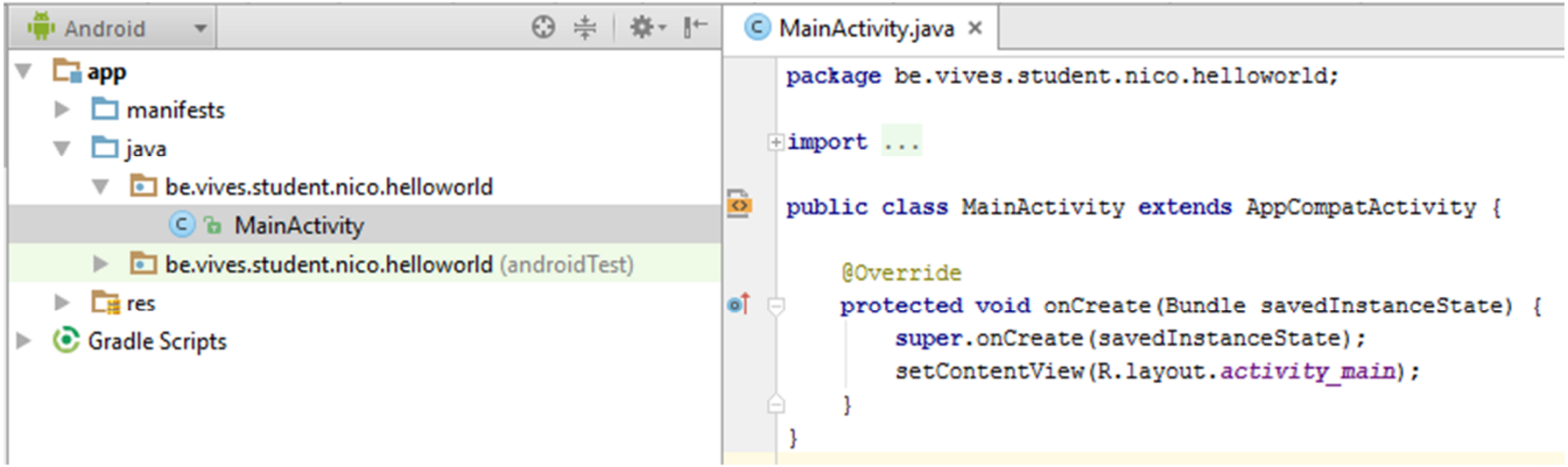
AndroidManifests.xml



Anatomy of an Android Application

Java Source Code Files

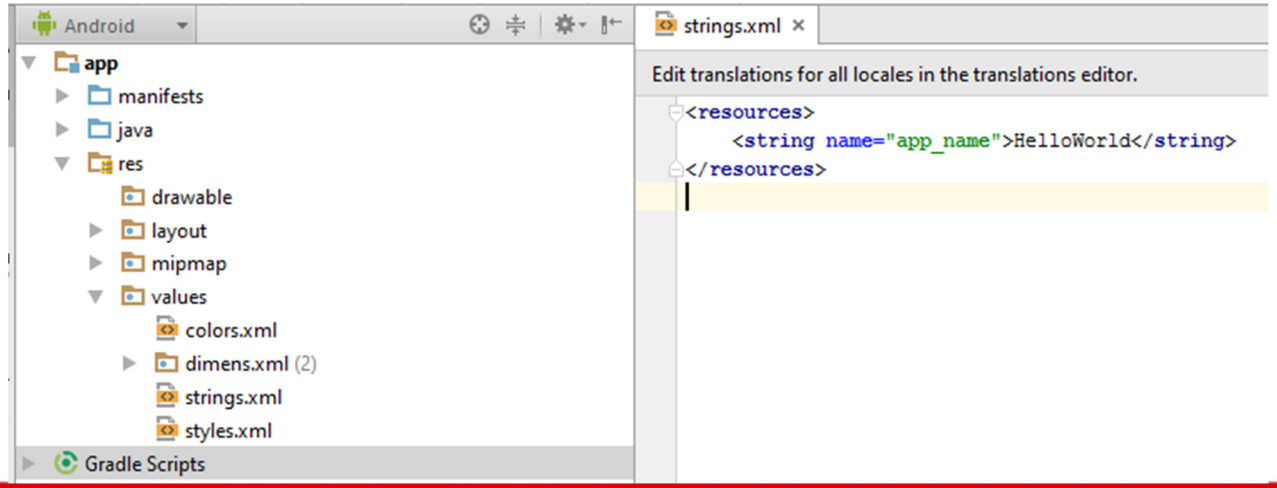
- In the java element you find all the source code packages and their included source code files
  - In this app we find the "MainActivity.java" source code file
    - Note that in the Android project view the extension is not shown

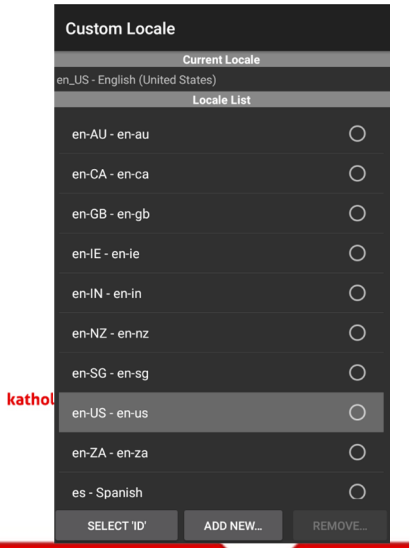


Anatomy of an Android Application

Resources

- The res element contains all the resource files used in the application such as
  - images (multiple resolutions)
  - GUI layout
  - string literals (strings.xml; selection based on users locale)
    - In computing, a locale is a set of parameters that defines the user's language, country and any special variant preferences that the user wants to see in their user interface.
    - To see which locale are supported open a AVT and launch the application "Custom Locale"

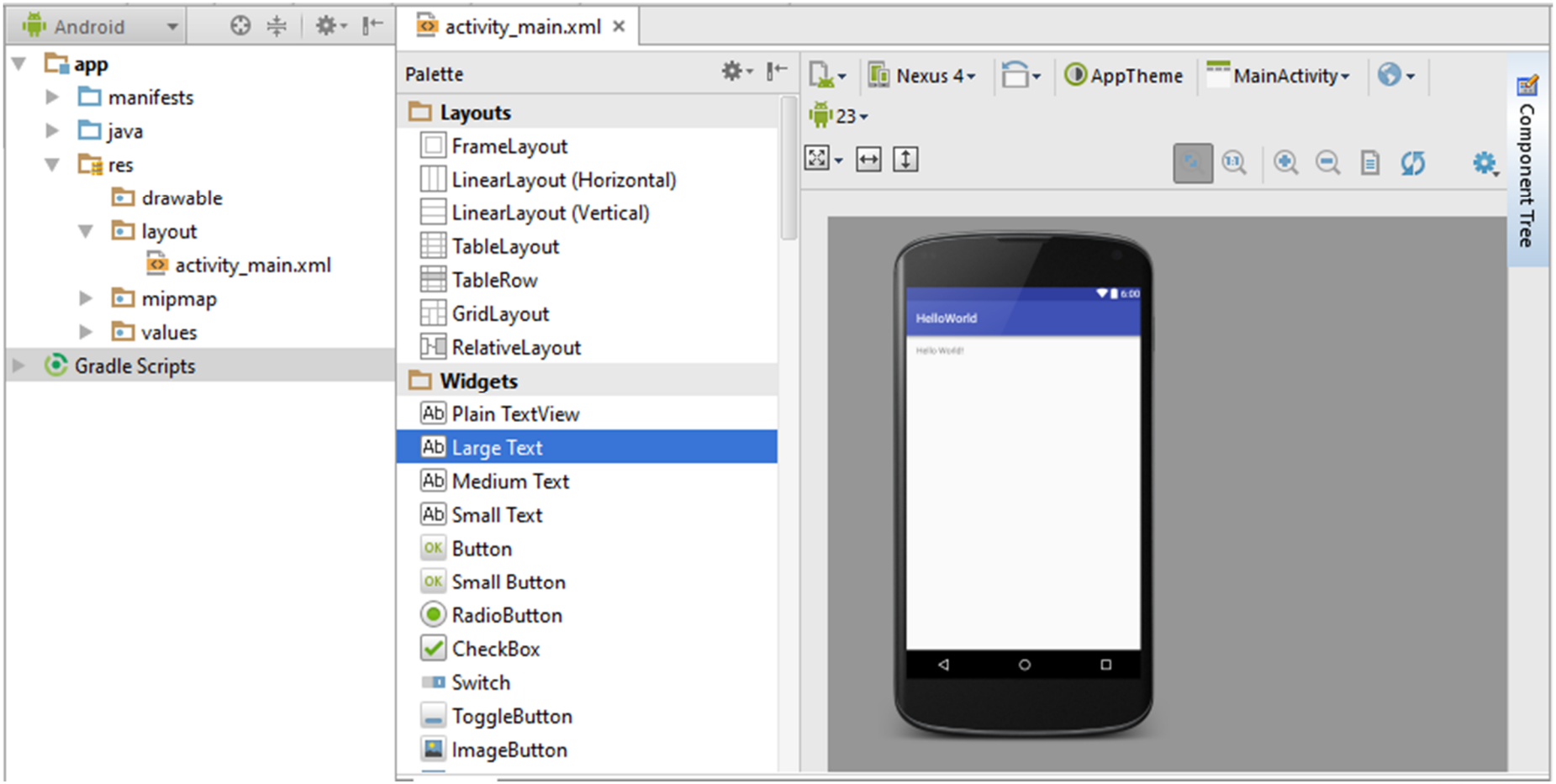




Anatomy of an Android Application

Resources

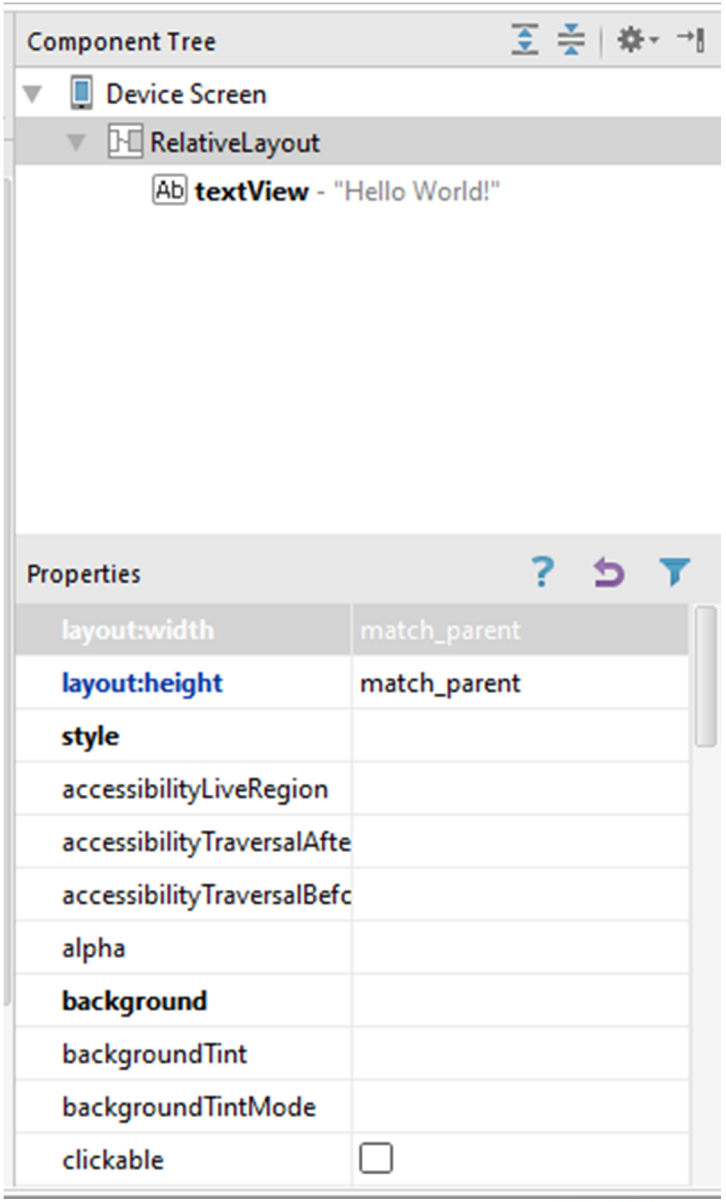
- res/layout/*.xml
  - Defines the user interface for your activities.



Anatomy of an Android Application

Resources

- You can edit visual component properties by opening the "Component Tree" pane on the right
- It will also show you a hierarchical view of the GUI components



Anatomy of an Android Application

Resources

- You can also directly edit the XML markup by selecting the "Text" view at the left bottom of the window