Tutorial 01

Introduction to Android – 01

SEG3125 – Analysis and Design of User Interfaces – Summer 2018

Presented by: Mohammad Al-hammouri– TA

Class Plan

- Introduction to Android
- Android Studio
  - Installation Guide
  - Project Setup
  - Android Simulation Setup
  - Interface Setup
- Basic Concepts
- UI Guideliness

INTRODUCTION

What's Android

- The world's most popular mobile platform:
  - *86.8%* market share.
- Powerful open development framework
  - Android Developer Tools offer a full Java IDE with advanced features for developing, debugging, and packaging Android apps
  - Not tied to any individual hardware manufacturers
- **Google Play:** Open marketplace to distribute your apps
  - One time *USD 25$* developer registration fee

Reference: http://www.idc.com/promo/smartphone-market-share/os

How does Android Work?

- Android runs on top of the Linux Kernel
- Android Applications are sandboxed within Virtual Machines
- Since *Kitkat (4.4)* the Runtime Environment for Android started using ART (Android RunTime), previously using Dalvik.
- Android Applications are based on ***Activities***
  - *Activities* should represent *"things"* you can do, like a screen or another functionality (e.g.: Search)
  - *Applications* usually have multiple *Activities*

Android Version History

Older Releases – Pre UI Integration

Cupcake	Donut	Eclair	Froyo	Gingerbread	Honeycomb
v1.5 API LVL 3	v1.6 API LVL 4	v2.0 – 2.1 API LVL 5–7	v2.2 – 2.2.3 API LVL 8	v2.3 – 2.3.7 API LVL 9–10	v3.0 – 3.2.6 API LVL 11–13

Newer Releases – Unified User Interface

Ice Cream Sandwich	Jelly Bean	KitKat	Lollipop	Marshmallow	Nougat
v4.0 – 4.2 API LVL 14–15	v4.1 – 4.3.1 API LVL 16–18	v4.4 – 4.4.4 API LVL 19–20	v5.0 – 5.1.1 API LVL 21–22	v6.0 API LVL 23	v7.0 – 7.1.1 API LVL 24

Each version of Android introduces new features.
Changes can focus on architecture changes, new features, optimizations or bug fixes.

Android Studio

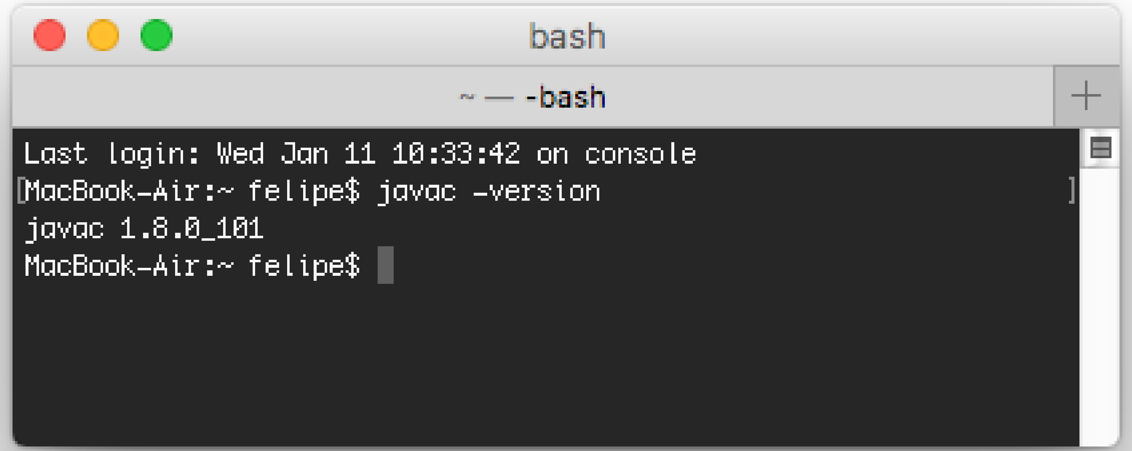
INSTALLATION

Installation

- Install Oracle JDK 8
  - Required for Android 5.0+
- Install Android Studio
  - Current Version: **2.2.3**
- *Optional* - Install an Alternate Emulator
  - Genymotion
  - BlueStacks

Java Development Kit (JDK)

1. Verify the current Java Installation:
   - Type *"**javac –version**"* on a terminal/console window to check your current installation.
   - If you have Java 7 or greater you are already set.
   - Otherwise, Proceed and Download the JDK.



```
Last login: Wed Jan 11 10:33:42 on console
MacBook-Air:~ felipe$ javac -version
javac 1.8.0_101
MacBook-Air:~ felipe$
```

Java Development Kit (JDK)

2. Download and install the appropriate version of the JDK:
   - The Current available build of the JDK8 is **8u111**
   - Your system will likely require the **64-bit** option.

- If you have multiple installations and are sure you have the minimum requirements, you can also proceed.

Android SDK

- The Android SDK provides access to the API libraries and developer tools necessary to build, test, and debug apps for Android.
- http://developer.android.com/sdk/index.html
- Available Tools
  - **Android Studio (IntelliJ IDEA)**
  - Command-Line Tools (Debug/Emulation)

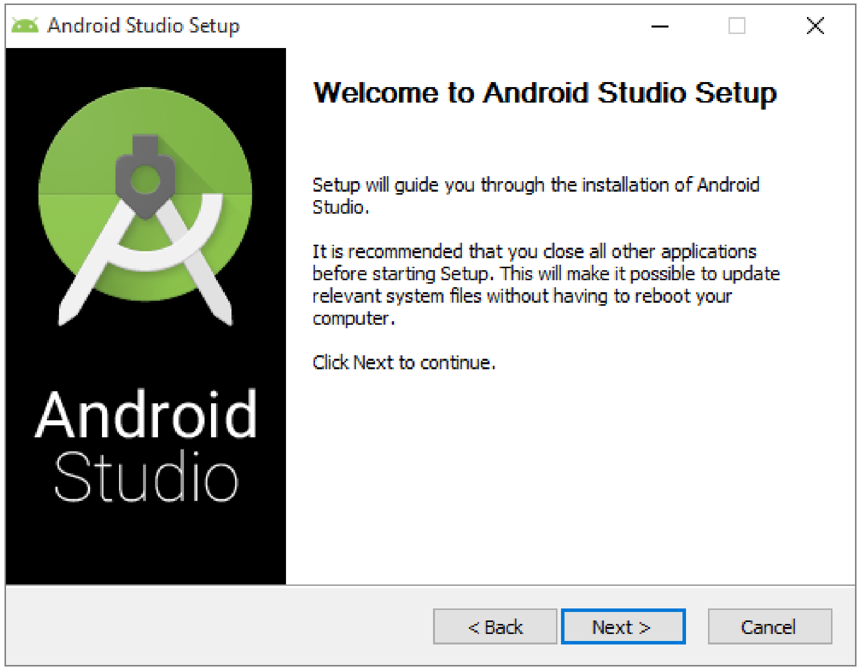
Android SDK

- Installing the Android SDK:
  - **Windows**: Follow Setup Wizard Instructions.
  - **Mac OS X**: Drag into Application Folder.
  - **Linux:** Download the .zip and follow instructions.
    - *Note: Using the OpenJDK may cause the installation to fail, use the linux variant of the Oracle JDK.*
  - *Additional Instructions Available here.*

- *Make sure your system has at least 3GB of free storage!*
  - Android System Images will require additional space.

Notes on Installation Process

- Install Android Studio
  - *Installation is straightforward. Just keep pressing next/accept*
  - *If multiple students attempt to download and install AS at the same time their downloads might become unbearably slow. Install prior to the lab if possible.*
- Select your UI
- Update the tools
  - Install Additional SDKs



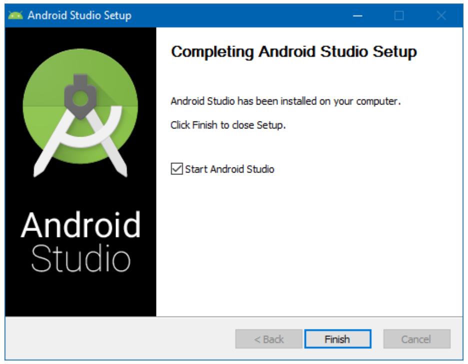
Welcome to Android Studio Setup

Setup will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Installation - Screenshots

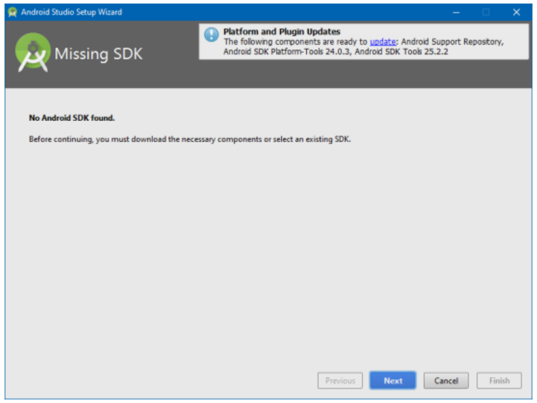


Completing Android Studio Setup

Android Studio has been installed on your computer.

Click Finish to close Setup.

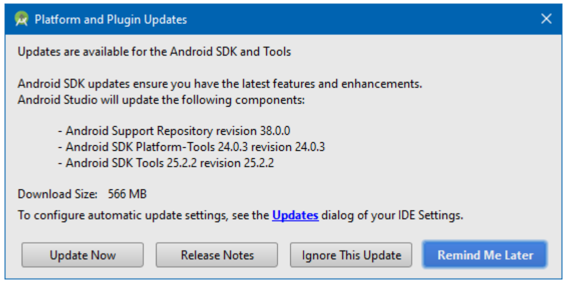
[✓] Start Android Studio



Platform and Plugin Updates
The following components are ready to update: Android Support Repository, Android SDK Platform-Tools 24.0.3, Android SDK Tools 25.2.2

No Android SDK found.

Before continuing, you must download the necessary components or select an existing SDK.



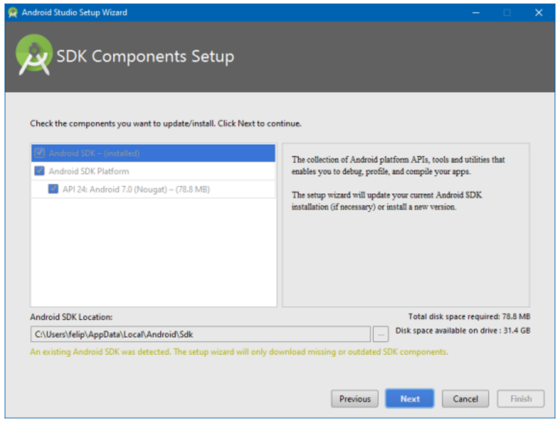
Updates are available for the Android SDK and Tools

Android SDK updates ensure you have the latest features and enhancements.
Android Studio will update the following components:

- Android Support Repository revision 38.0.0
- Android SDK Platform-Tools 24.0.3 revision 24.0.3
- Android SDK Tools 25.2.2 revision 25.2.2

Download Size: 566 MB

To configure automatic update settings, see the Updates dialog of your IDE Settings.



Check the components you want to update/install. Click Next to continue.

[✓] Android SDK – (installed)
[✓] Android SDK Platform
[✓] API 24: Android 7.0 (Nougat) – (78.8 MB)

The collection of Android platform APIs, tools and utilities that enables you to debug, profile, and compile your apps.

The setup wizard will update your current Android SDK installation (if necessary) or install a new version.

Android SDK Location: C:\Users\felip\AppData\Local\Android\Sdk

Total disk space required: 78.8 MB
Disk space available on drive : 31.4 GB

An existing Android SDK was detected. The setup wizard will only download missing or outdated SDK components.

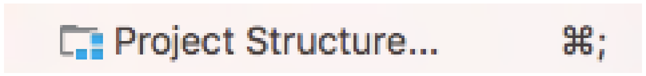
Android SDK: Multiple JDKs

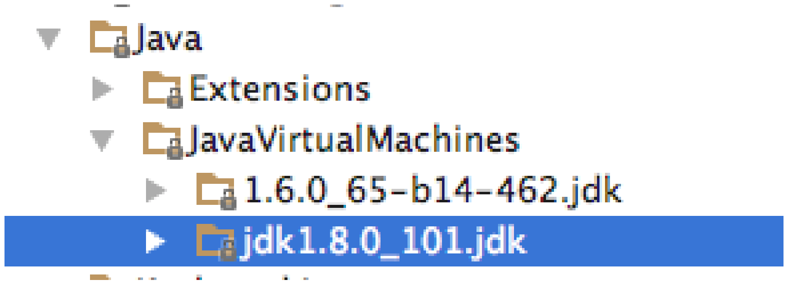
In certain scenarios where multiple versions of the JDK are installed in the same system, Android Studio might select an older version of the JDK during setup. To fix this issue:

1. Go to: *File >> Project Structure >> SDK Location*, and switch the version of the JDK used.

2. Select either (a) a *jdk1.8* variant or (b) the option "Use Embedded JDK" if available.

3. Restart Android Studio.





Multiple JDKs – MAC OS X

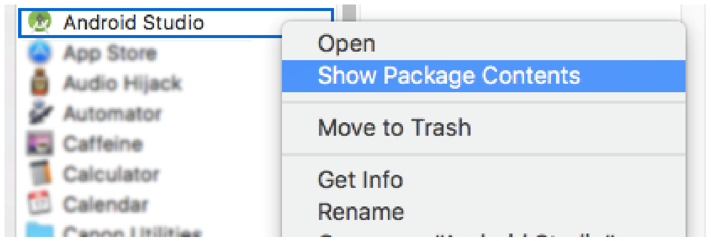
If you have a **Mac OS X** computer and updating the JDK in your project configurations does not fix existing issues (warnings or other JDK related compilation issues):

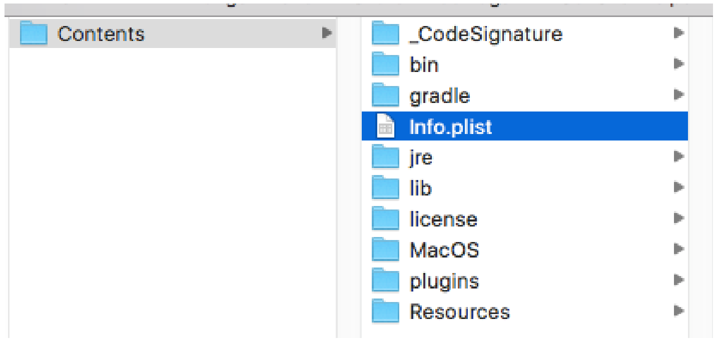
1. Close Android Studio and go to the folder where you installed it (Applications Folder)

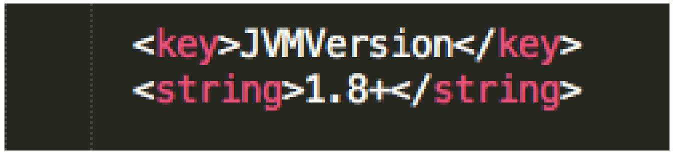
2. Right-click on the app icon and Select "Show Package Contents"

3. Navigate to: **"Contents >> Info.plist"** and open it with a text editor *(e.g.: Sublime Text)*

4. Edit the line below the JVMVersion key so it specifies **Version 1.8** (See Image to the side)







```
<key>JVMVersion</key>
<string>1.8+</string>
```

Android Studio: Caveats

- Certain Computers do not cope well with Android Studio:
  - Older Computers: Java Runtime Rendering Errors
  - Older Microsoft Surface Pro devices: General Interface Limitations
  - QHD & 4K Monitors: UI Scaling Issues can happen!
- Android Studio may crash during execution. If you get Rendering Errors or other stability issues, restart Android Studio.

Android Studio

TEST DEVICES

Testing your Projects

- Using an Android Phone
  - **Instant Run:** Allows changes to be pushed to existing app installation without need for a new build
    - *Requires the deployment target to be the same version as the device.*
    - *Most phones run older versions of Android, so building for nougat will not allow for Instant Run*

- Using an Emulator
  - Android Studio includes an Emulator
  - Other Emulators can be installed

Using a Test Device

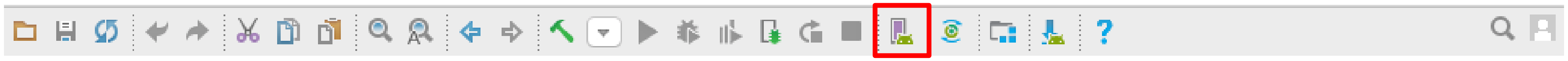
- Windows:
  - Install OEM USB Drivers to use an Android Phone for Debugging.
- Mac OS:
  - Install Android File Manager to access/move files on the device.
- **Enable USB debugging on your device.**
  - On most devices running Android 3.2 or older, you can find the option under **Settings > Applications > Development**.
  - On Android 4.0 and newer, it's in **Settings > Developer options**.

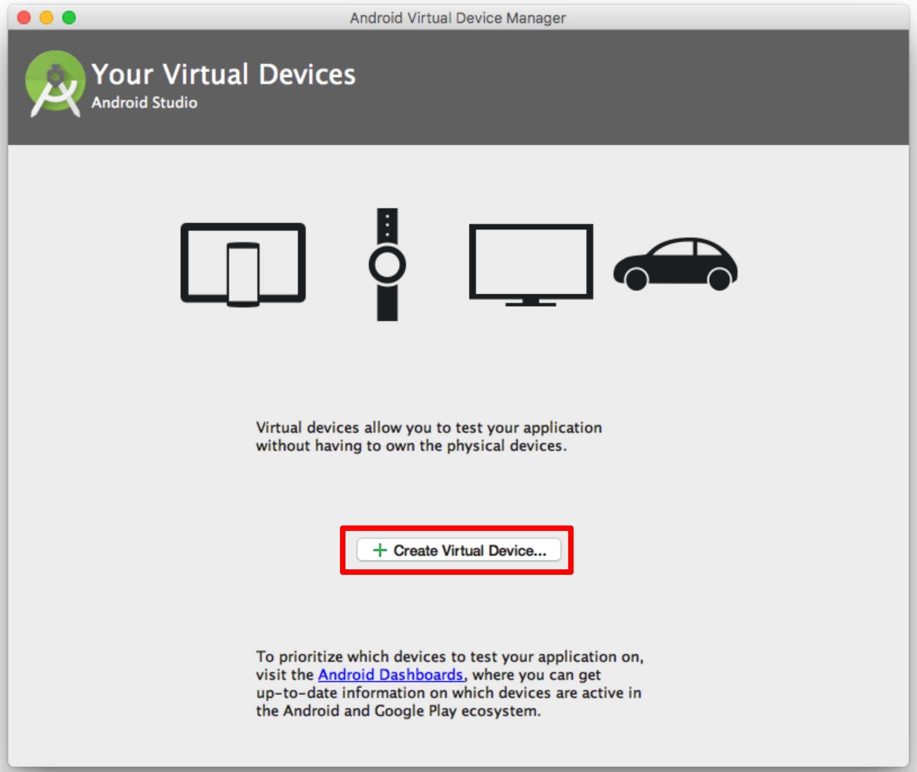
Note: *On Android 4.2 and newer,* ***Developer options*** *is hidden by default. To make it available, go to* ***Settings > About phone*** *and tap* ***Build number*** *seven times. Return to the previous screen to find* ***Developer options****.*

Using an Android Virtual Device (AVD)

- An Android Virtual Device (AVD) is an emulator configuration that lets you model an actual device. It consists of :
  - **A hardware profile:** e.g., whether the device has a camera
  - **A mapping to a system image:** e.g. which Android version
  - **A dedicated storage area on your development machine:** e.g. the device's user data
  - **Other options:** e.g. the emulator skin, appearance, and so on

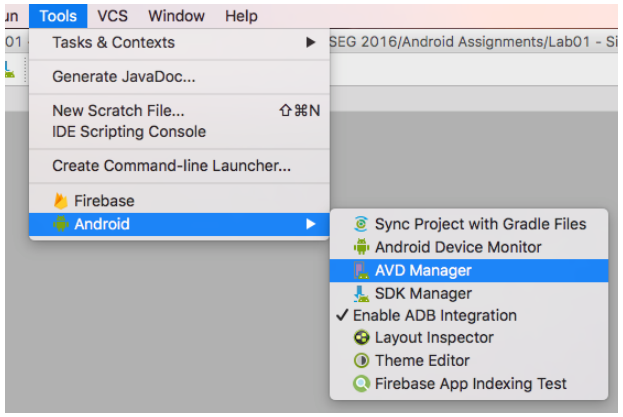
Creating an AVD





Virtual devices allow you to test your application without having to own the physical devices.

To prioritize which devices to test your application on, visit the Android Dashboards, where you can get up-to-date information on which devices are active in the Android and Google Play ecosystem.



From the Menu Bar, select ***Tools > AVD Manager***,

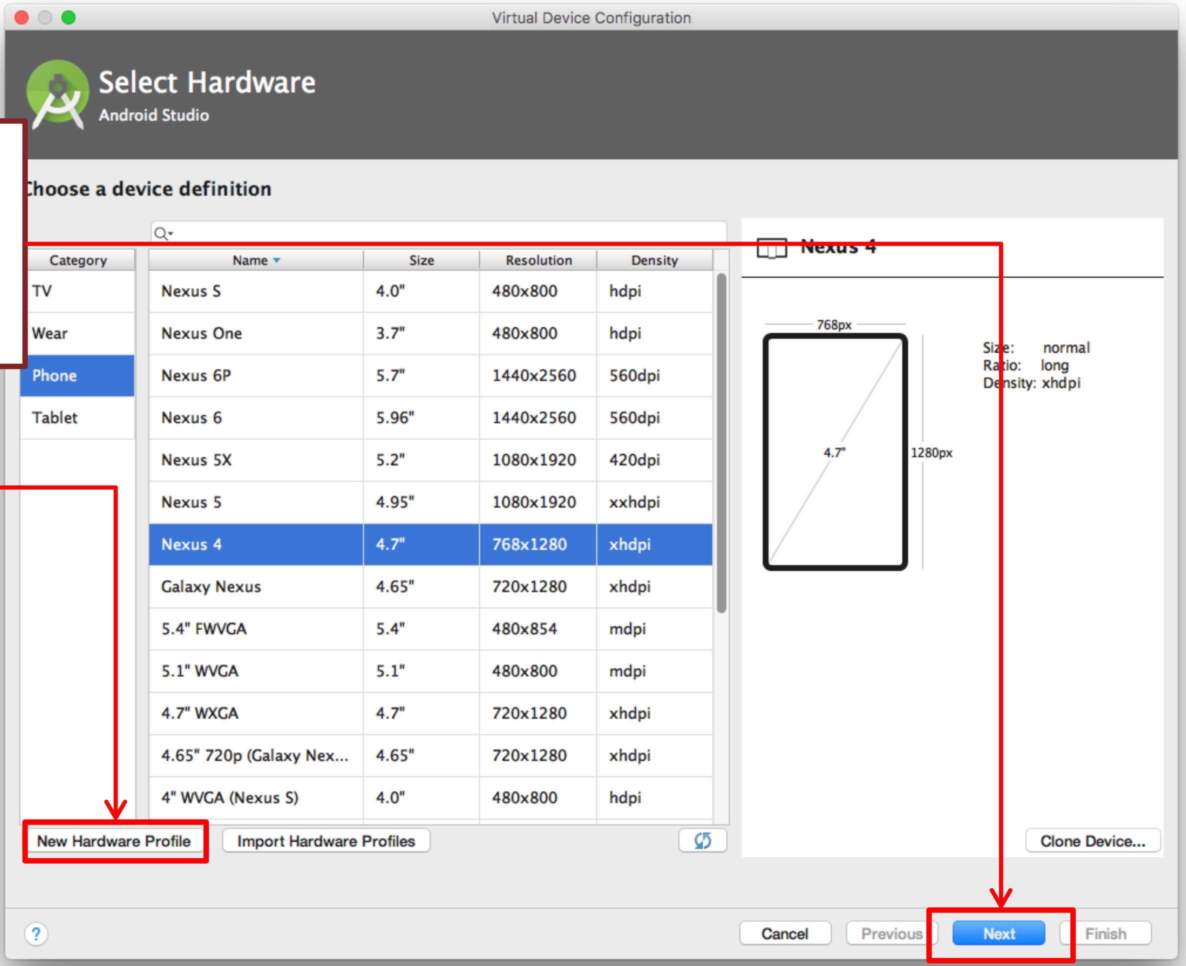
Alternatively, Click the **AVD Manager** icon in the toolbar

Click the "**+**" button to create a new AVD

Creating an AVD

Select a device from the preset list and press **"Next"** or create your own by clicking the **"New Hardware Profile"** Button.

Note: To reduce the strain on the computer running the AVD, we recommend students use devices with less demanding requirements, such as the Nexus 4.



Virtual Device Configuration

Select Hardware
Android Studio

Choose a device definition

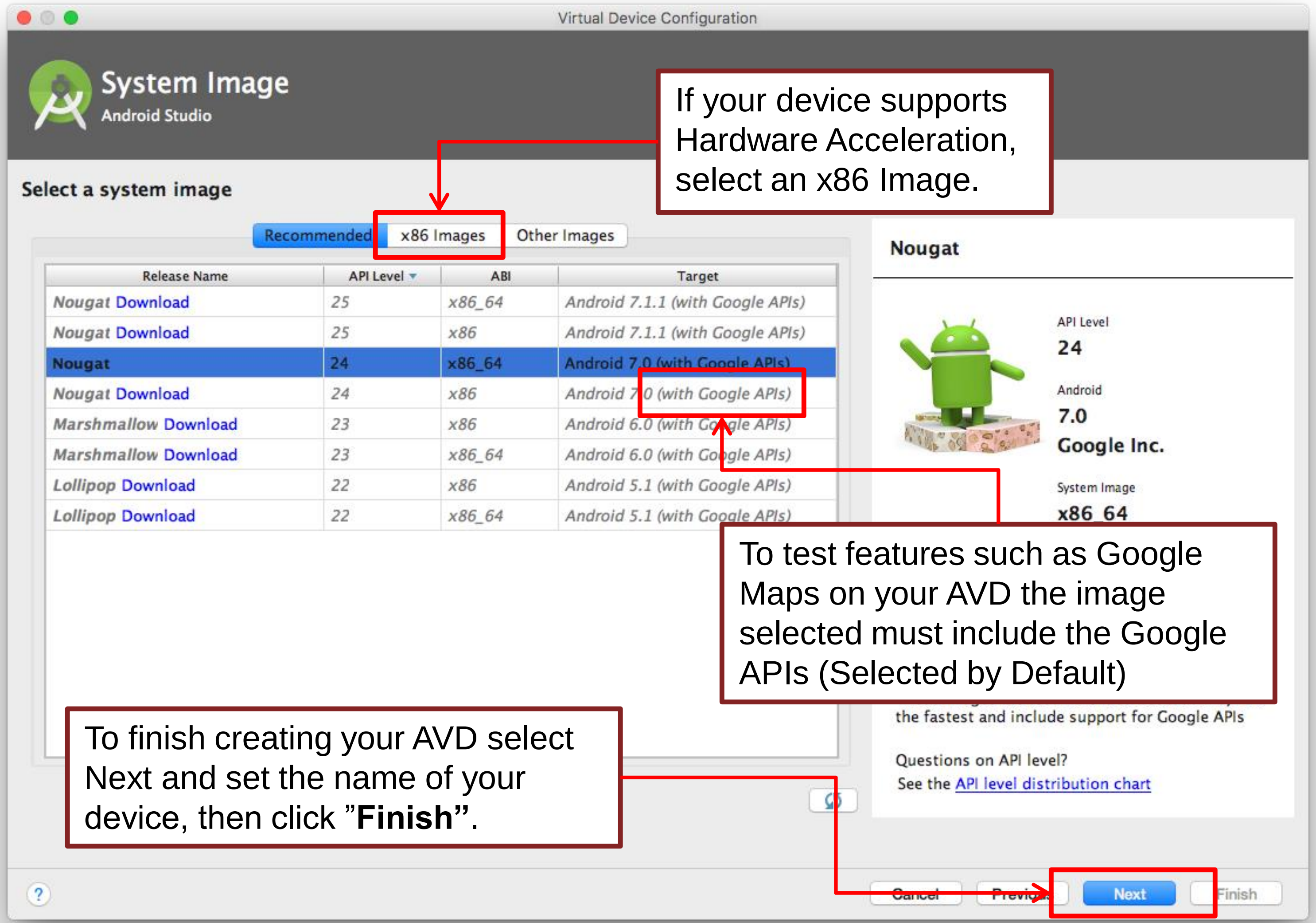
Category
TV
Wear
Phone
Tablet

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

Nexus 4

Size: normal
Ratio: long
Density: xhdpi

Creating an AVD



Virtual Device Configuration

System Image
Android Studio

Select a system image

Release Name	API Level	ABI	Target
Nougat Download	25	x86_64	Android 7.1.1 (with Google APIs)
Nougat Download	25	x86	Android 7.1.1 (with Google APIs)
Nougat	24	x86_64	Android 7.0 (with Google APIs)
Nougat Download	24	x86	Android 7.0 (with Google APIs)
Marshmallow Download	23	x86	Android 6.0 (with Google APIs)
Marshmallow Download	23	x86_64	Android 6.0 (with Google APIs)
Lollipop Download	22	x86	Android 5.1 (with Google APIs)
Lollipop Download	22	x86_64	Android 5.1 (with Google APIs)

Nougat

API Level
24

Android
7.0
Google Inc.

System Image
x86_64

the fastest and include support for Google APIs

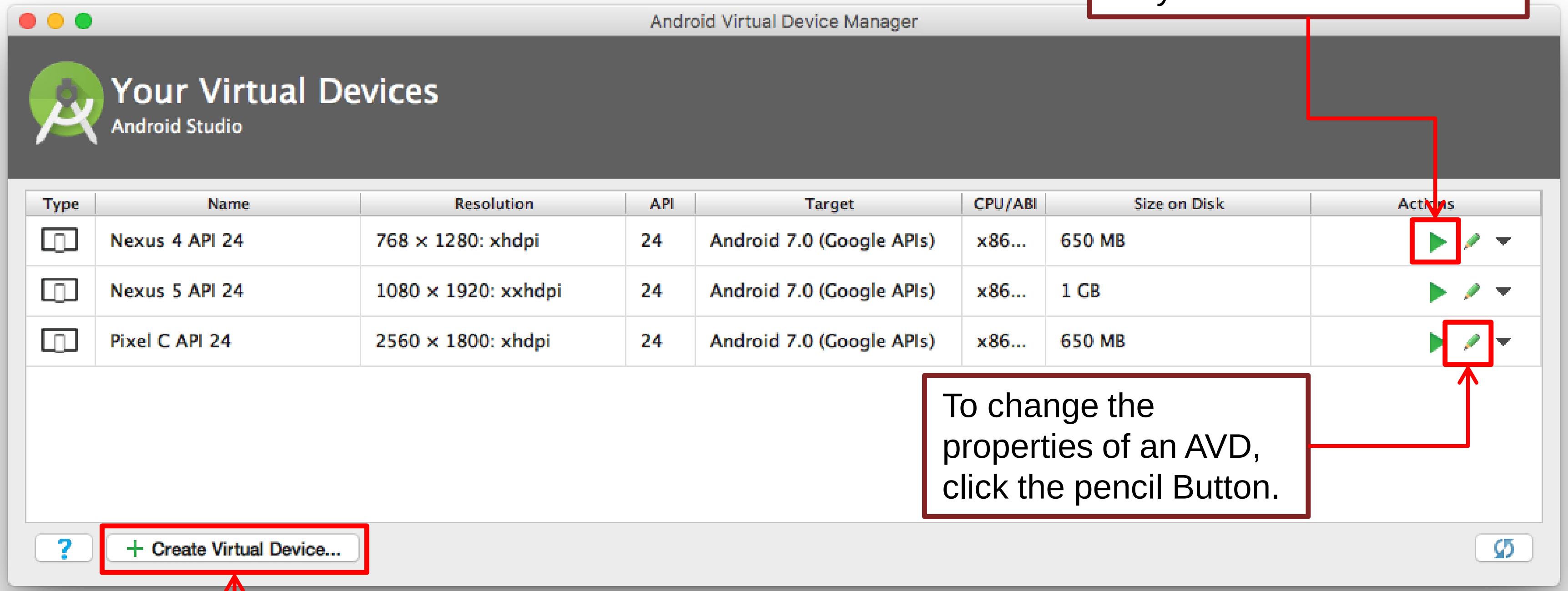
Questions on API level?
See the API level distribution chart

If your device supports Hardware Acceleration, select an x86 Image.

To test features such as Google Maps on your AVD the image selected must include the Google APIs (Selected by Default)

To finish creating your AVD select Next and set the name of your device, then click "**Finish**".

Starting an AVD



Android Virtual Device Manager

Your Virtual Devices
Android Studio

Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 4 API 24	768 × 1280: xhdpi	24	Android 7.0 (Google APIs)	x86...	650 MB	
	Nexus 5 API 24	1080 × 1920: xxhdpi	24	Android 7.0 (Google APIs)	x86...	1 GB	
	Pixel C API 24	2560 × 1800: xhdpi	24	Android 7.0 (Google APIs)	x86...	650 MB	

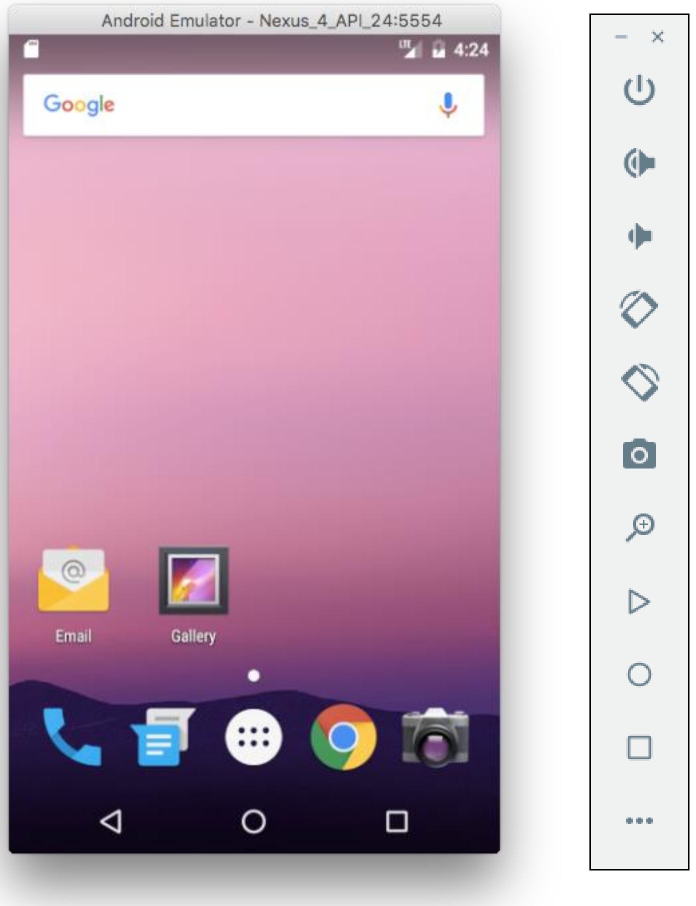
To Start an AVD click the Play Button.

To change the properties of an AVD, click the pencil Button.

You can create multiple devices to test your application in different versions of Android or on devices with different screen properties.

Android Emulator

- Once you start your AVD you do not have to close it when reloading your application.
- If your computer can perform while the AVD is running, keep it on to speed things up.
- When you recompile your code or re-run your application, it will be brought to the front on the test device being used.



Emulation: Caveats

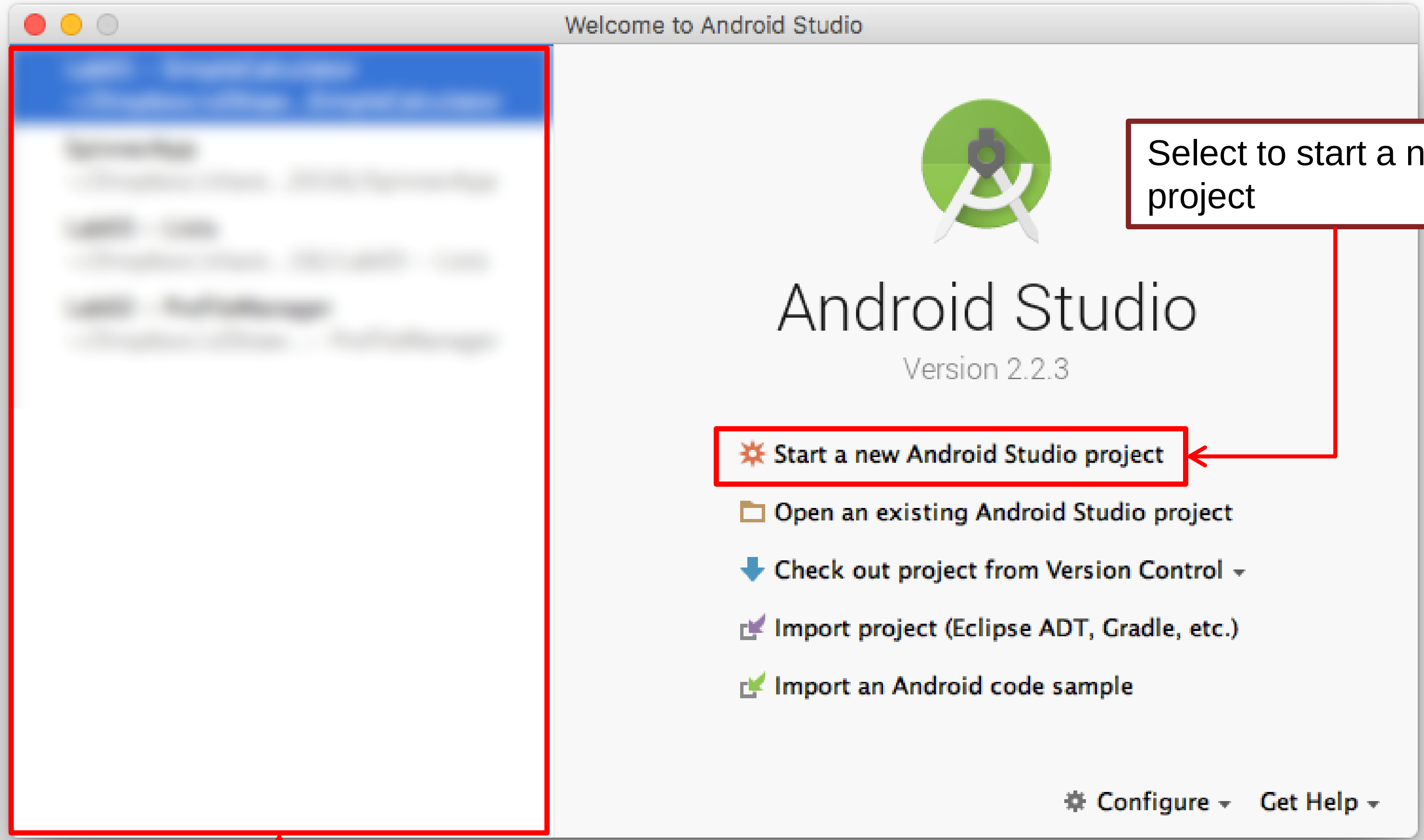
- Certain Computers will ***LAG*** when running android emulation:
  - *AMD CPUs: No Intel Hardware Acceleration*
  - *Tablet PCs: Limited Processing – Slow Emulation*

- x86 Device Emulation requires Intel HAXM:
  - Install HAXM if you get Emulator Warnings
  - Enable Hardware Acceleration in BIOS

- Good Emulating Alternatives: Genymotion, BlueStacks

Android Studio

CREATING A PROJECT (LIVE DEMO)


Welcome to Android Studio

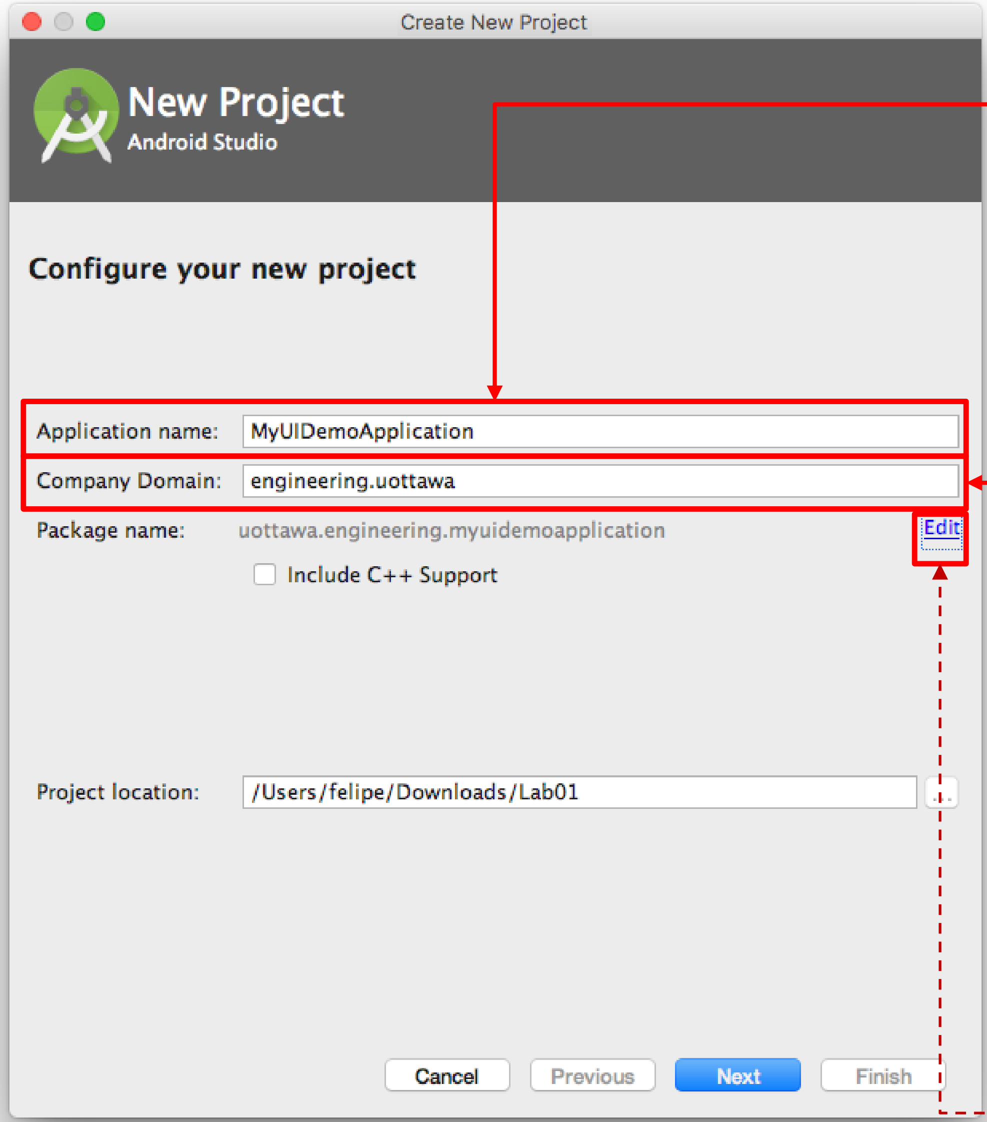
Android Studio

Version 2.2.3

- 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

Select to start a new project

Existing Projects



Create New Project

New Project
Android Studio

Configure your new project

Application name: MyUIDemoApplication
Company Domain: engineering.uottawa
Package name: uottawa.engineering.myuidemoapplication
Include C++ Support
Project location: /Users/felipe/Downloads/Lab01

Project Name viewed on Project List. Name should bew Unique!

Domain of the group developing the app

Note: Please remember to give your Projects and Activities meaningful names!

The developer domain is generally based on reverse .com domain hierarchy.

[country code].[top level domain].[business name].[subdomain].[team]

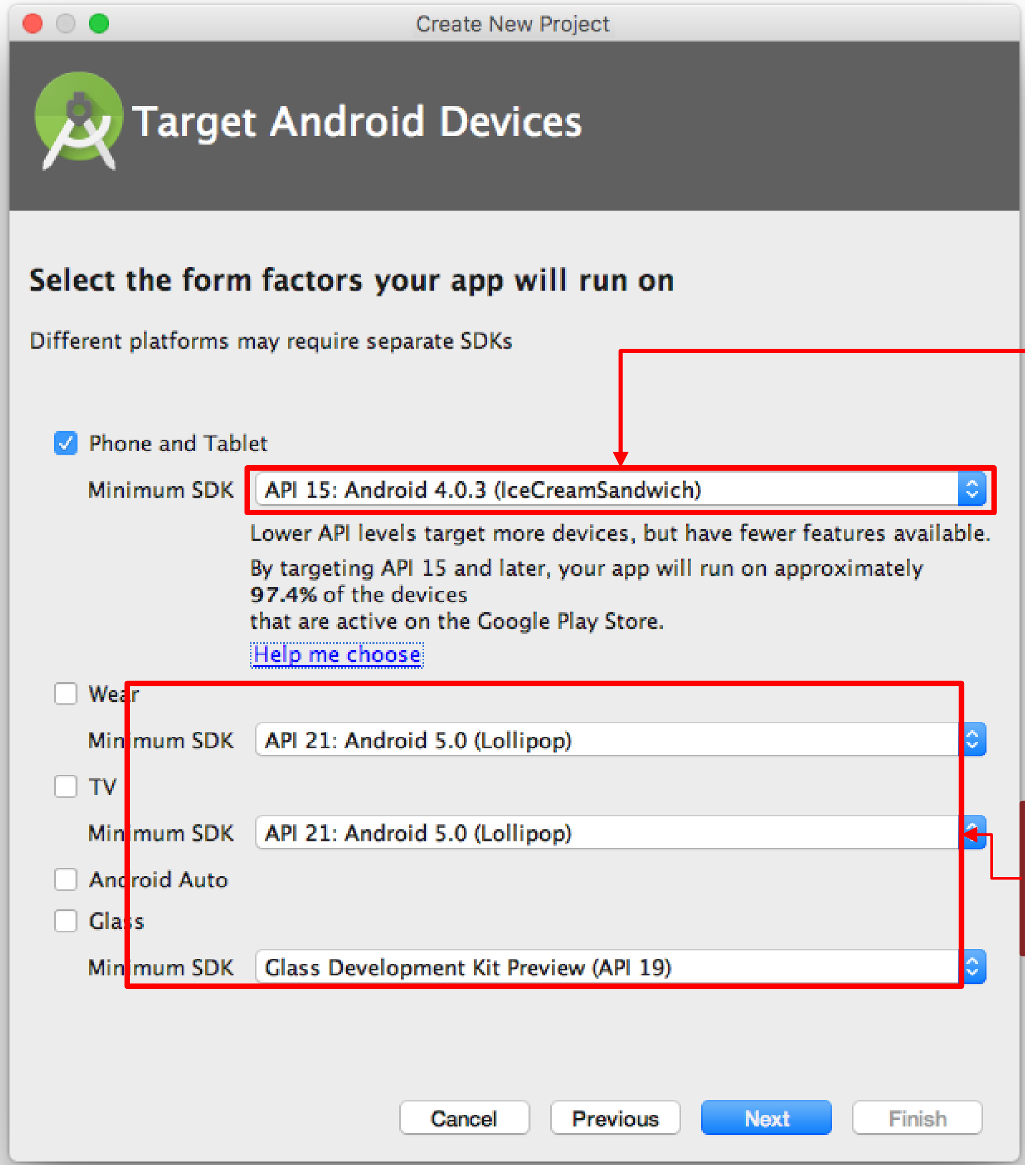
Ex: br.com.firasoft.msp.jimmyfive

- Country Code, Subdomain and Team Fields are optional, but help with structure.

The package name is produced from **Application Name** and **Company Domain.**

You can edit the Package Name if necessary

Note: Project locations should not contain whitespaces to avoid issues with the NDK Tools!



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 15: Android 4.0.3 (IceCreamSandwich)

Lower API levels target more devices, but have fewer features available.
By targeting API 15 and later, your app will run on approximately **97.4%** 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 21: Android 5.0 (Lollipop)

☐ Android Auto

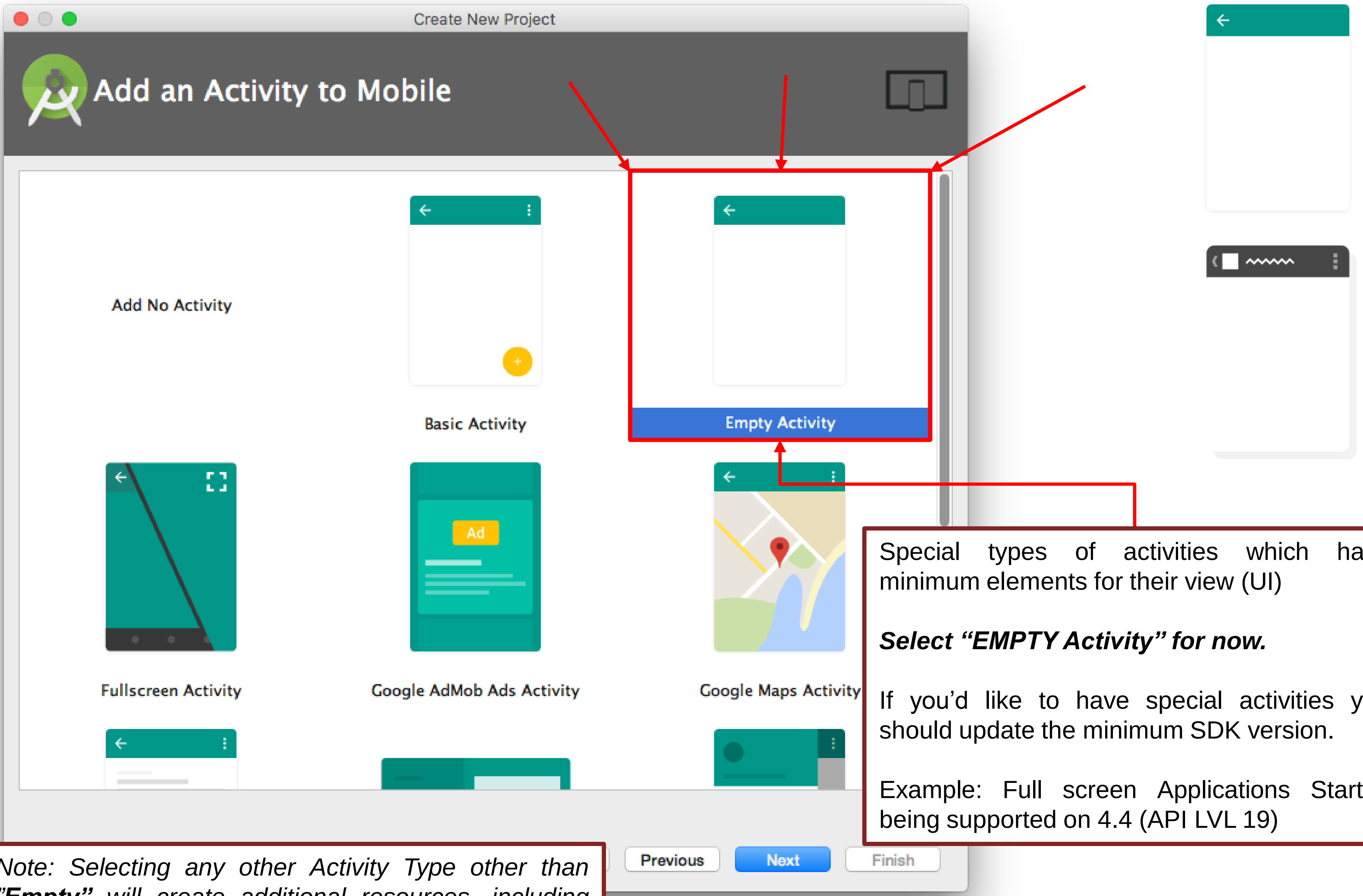
☐ Glass

Minimum SDK: Glass Development Kit Preview (API 19)

Minimum Version of Android

Standard Value is API LVL 15, we recommend the default value be kept.

After learning the ropes, you can develop and publish apps for multiple variations of the Android OS.



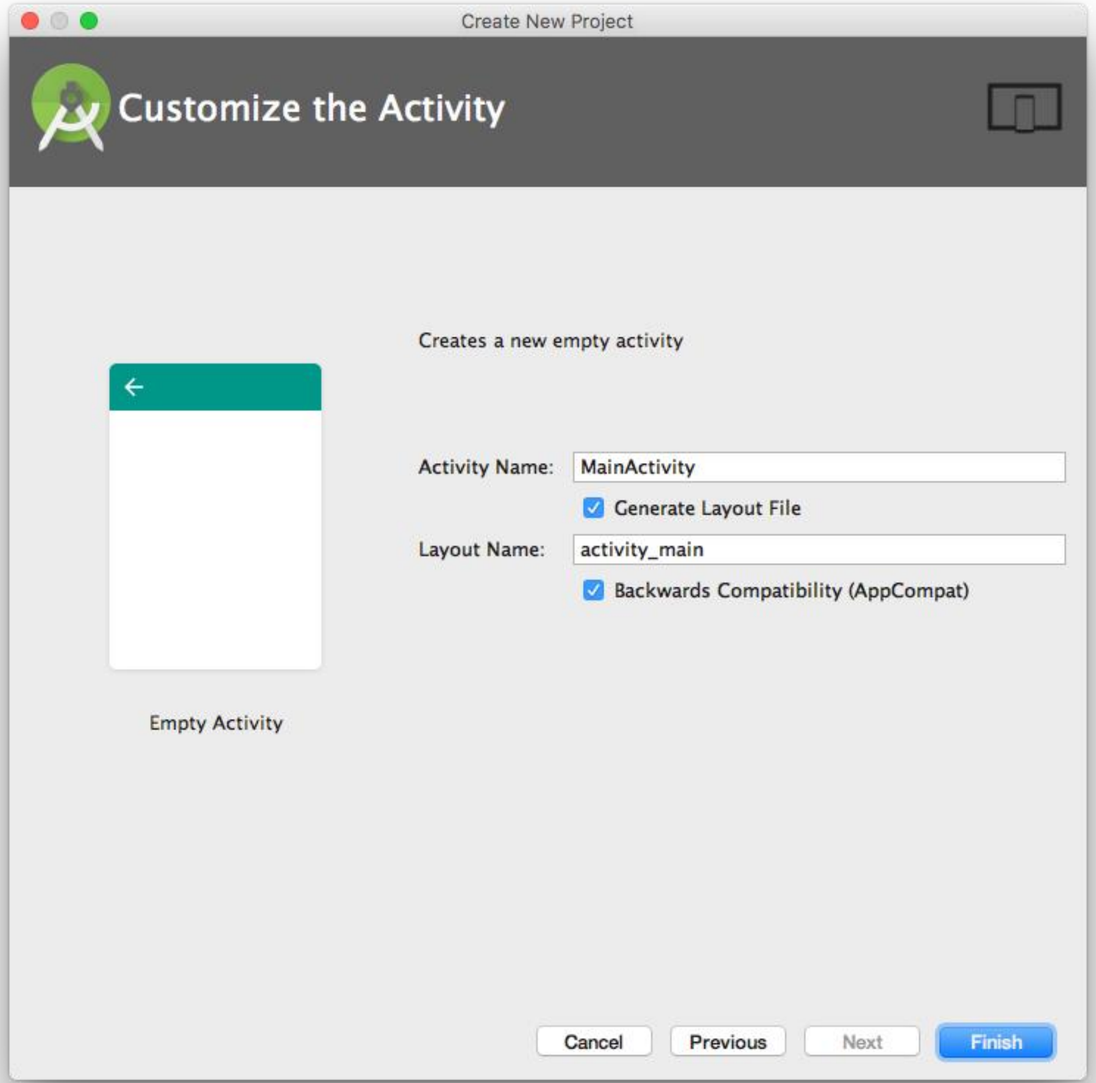
Special types of activities which have minimum elements for their view (UI)

Select "EMPTY Activity" for now.

If you'd like to have special activities you should update the minimum SDK version.

Example: Full screen Applications Started being supported on 4.4 (API LVL 19)

*Note: Selecting any other Activity Type other than "**Empty**" will create additional resources, including XML files and Java code.*


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)

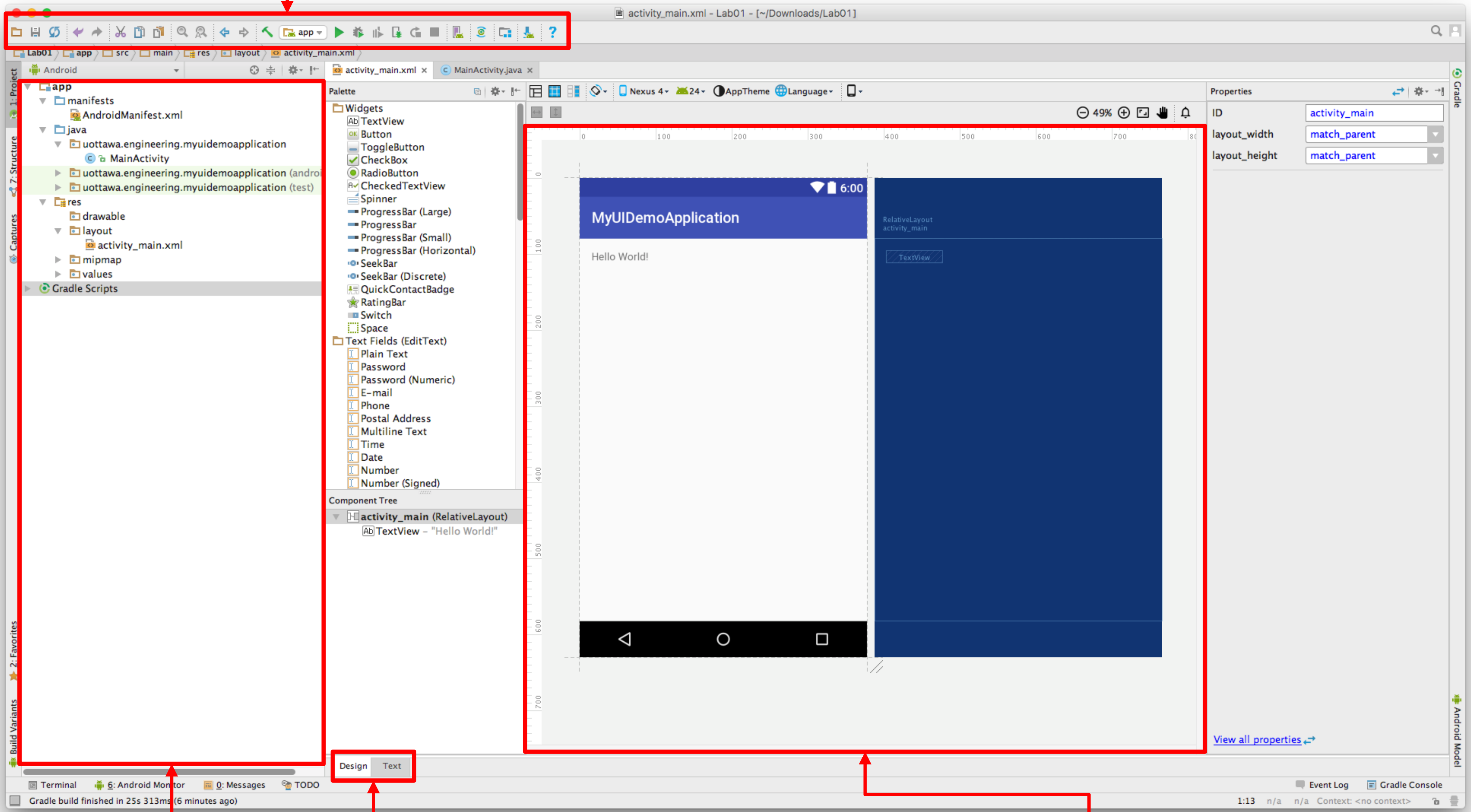
The **AppCompatActivity** allows access to the **ActionBar**

In this screen you can configure the naming scheme of your Activity, these details only affect the developer side of things.

By default Java uses Capitalized Camel Code as a standard for Classes, Interfaces and Camel Code for Methods and Variables. Check Oracle's website for more details:

- *http://www.oracle.com/technetwork/java/codeconventions-135099.html*

Toolbar



Project View

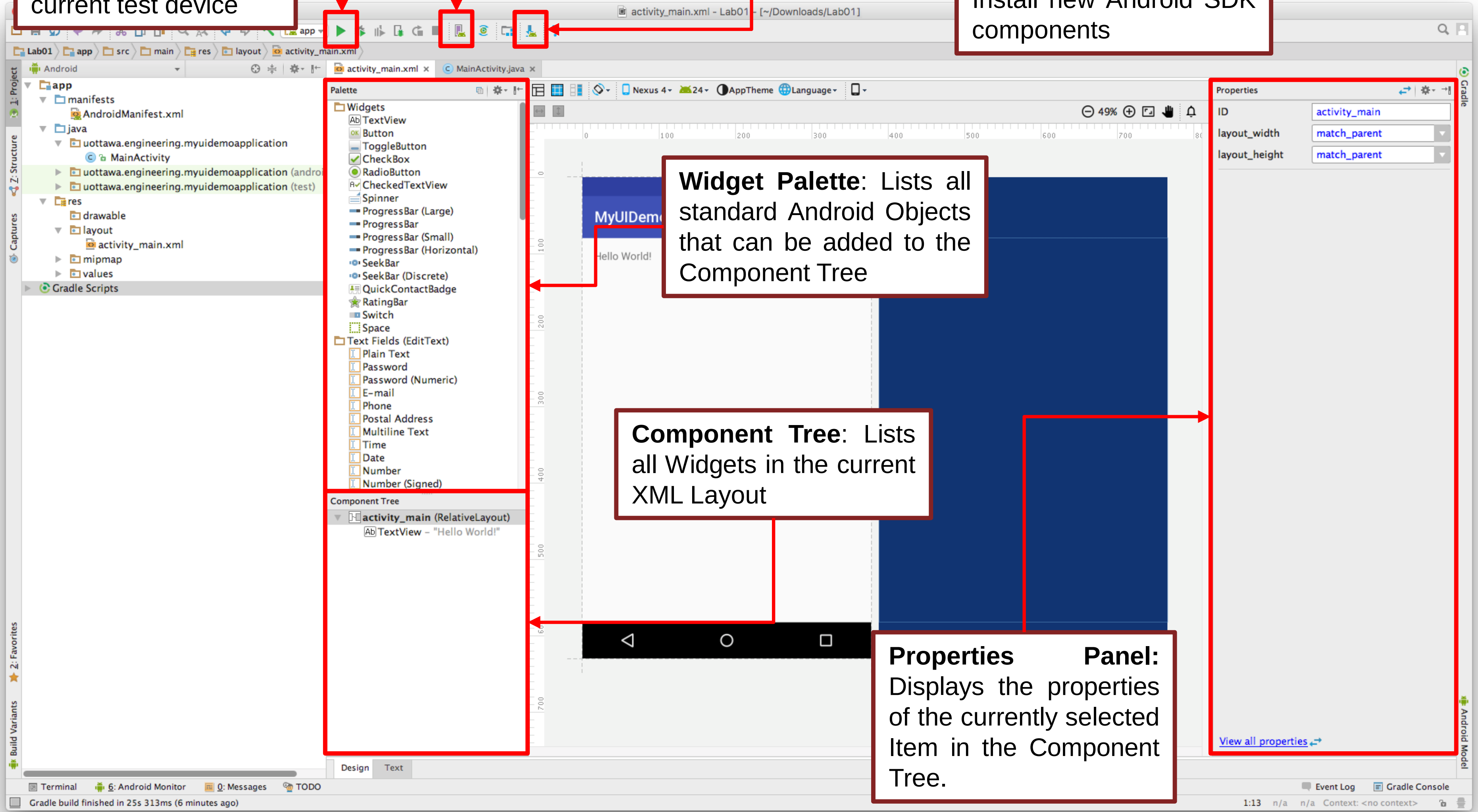
Toggle between Design Preview and XML Source code

Design View: Visual Preview of contents in Layout XML file

Play Button: Runs the Project on the current test device

AVD Button: Opens the Virtual Device Manager.

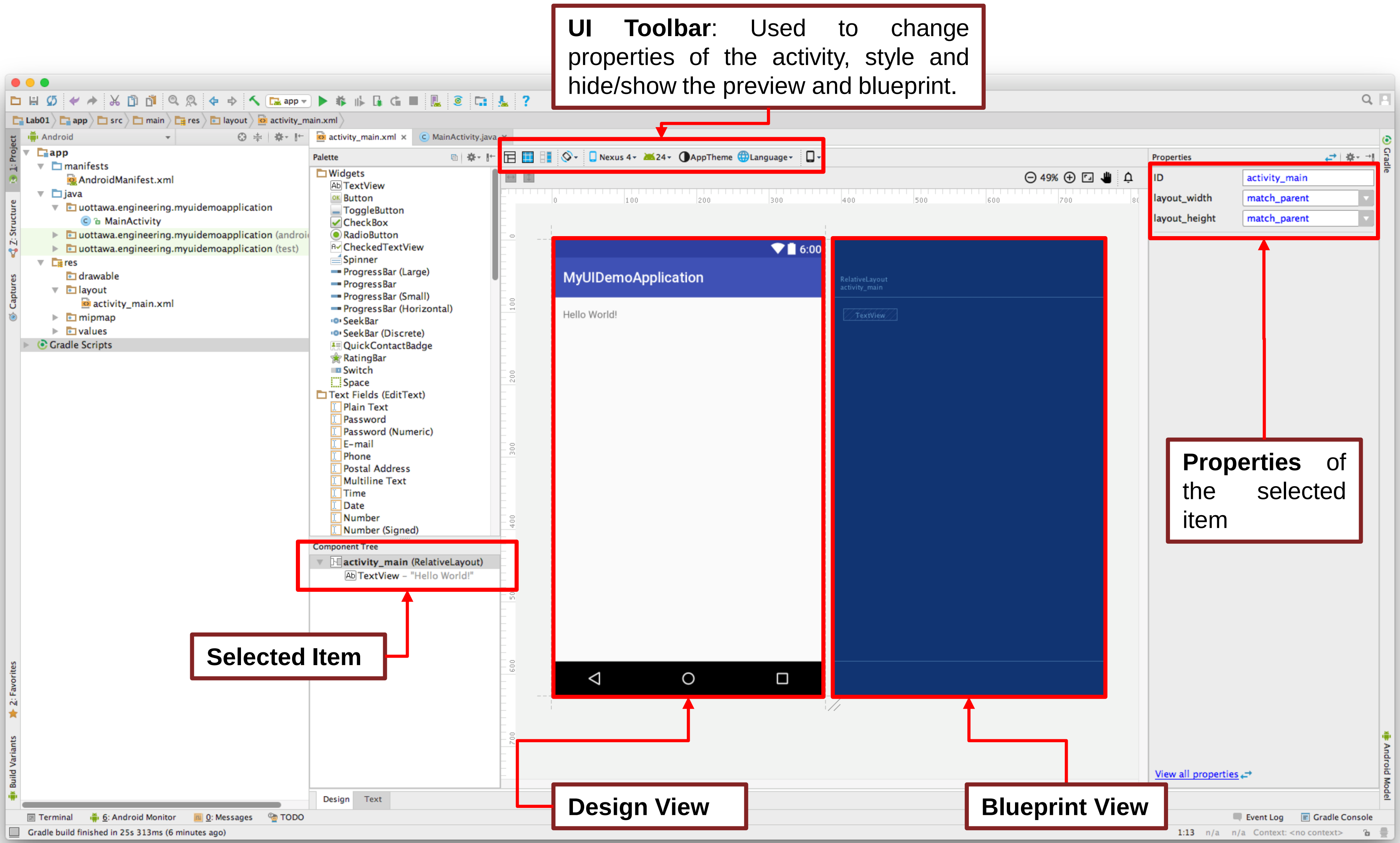
SDK Button: Opens the screen where you can Install new Android SDK components



Widget Palette: Lists all standard Android Objects that can be added to the Component Tree

Component Tree: Lists all Widgets in the current XML Layout

Properties Panel: Displays the properties of the currently selected Item in the Component Tree.



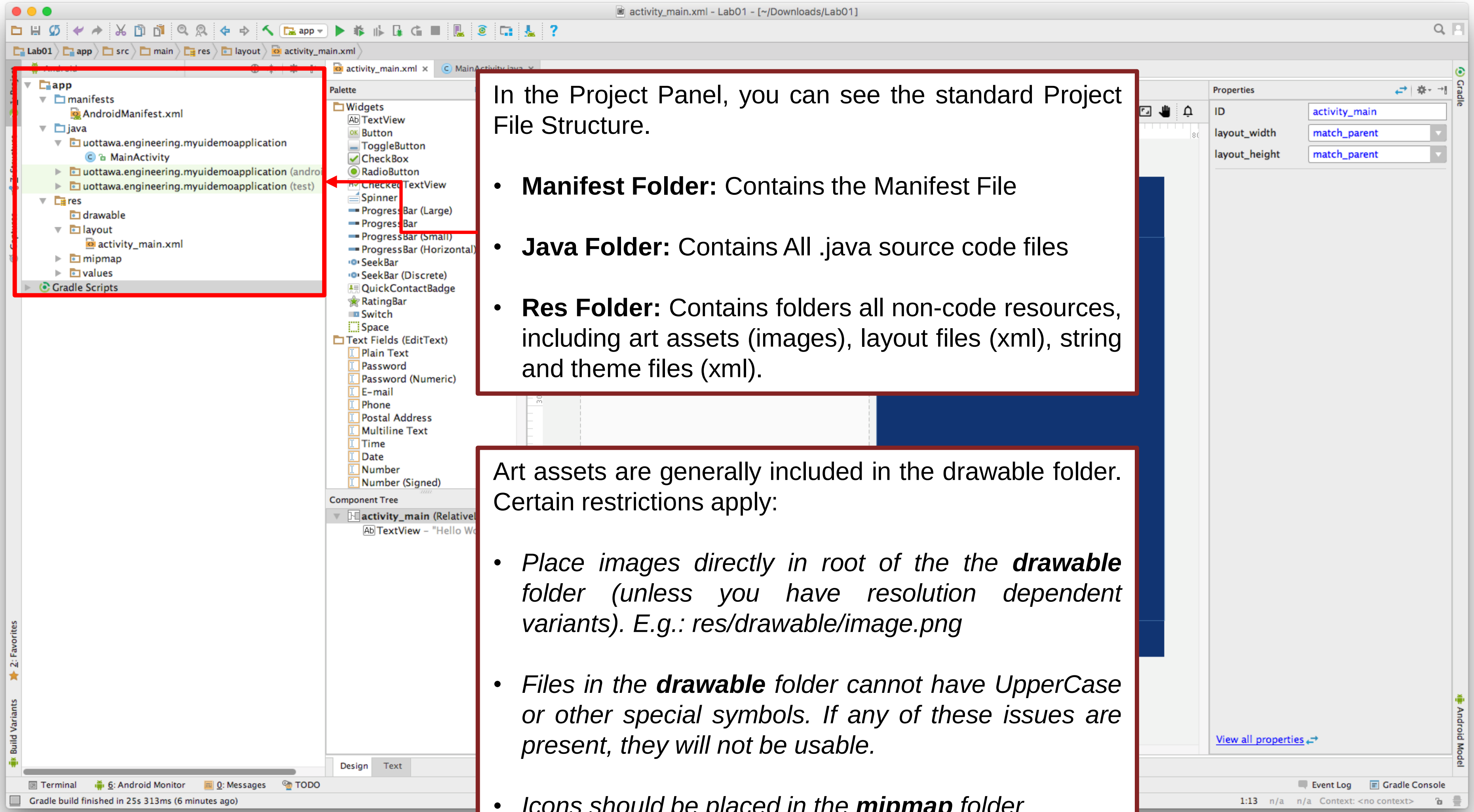
UI Toolbar: Used to change properties of the activity, style and hide/show the preview and blueprint.

Properties of the selected item

Selected Item

Design View

Blueprint View



In the Project Panel, you can see the standard Project File Structure.

- **Manifest Folder:** Contains the Manifest File
- **Java Folder:** Contains All .java source code files
- **Res Folder:** Contains folders all non-code resources, including art assets (images), layout files (xml), string and theme files (xml).

Art assets are generally included in the drawable folder. Certain restrictions apply:

- *Place images directly in root of the the **drawable** folder (unless you have resolution dependent variants). E.g.: res/drawable/image.png*
- *Files in the **drawable** folder cannot have UpperCase or other special symbols. If any of these issues are present, they will not be usable.*
- *Icons should be placed in the **mipmap** folder*

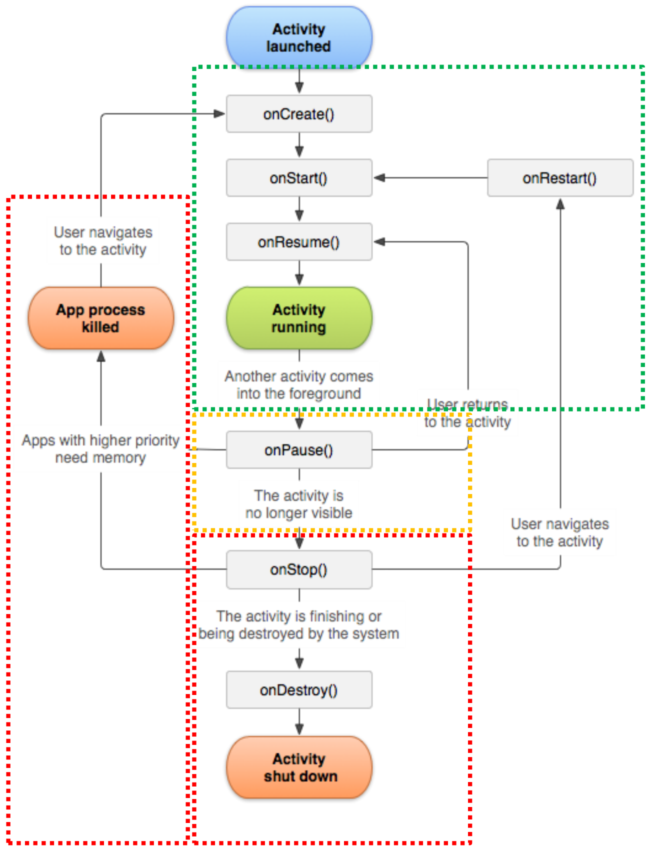
Android Manifest

- File containing the basic configuration of the of Android application.
- Includes: App Name, App Package, Used Theme, Target Version, Intents, Activities.
- All the elements that can appear in the manifest file are listed in alphabetical order. These are the only legal elements; you cannot add your own elements or attributes.

Activity

- There is no **Main()**, activities are started from their **OnCreate()**, with the first activity creation being called by the Android System.
- "An ***Activity*** is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map."
- An application usually consists of multiple activities that are loosely bound to each other, with one being the ***Main activity***.
- Individual Activities perform specific tasks and are connected by ***Intents***.
- Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack.

Activity


Activity launched
onCreate()
onStart()
onRestart()
onResume()
Activity running
Another activity comes into the foreground
User returns to the activity
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down
User navigates to the activity
App process killed
Apps with higher priority need memory

The **entire lifetime** of an activity happens between the first call to **onCreate()** through to a single final call to **onDestroy()**.

An Activity can be Active, in the Foreground or Destroyed.

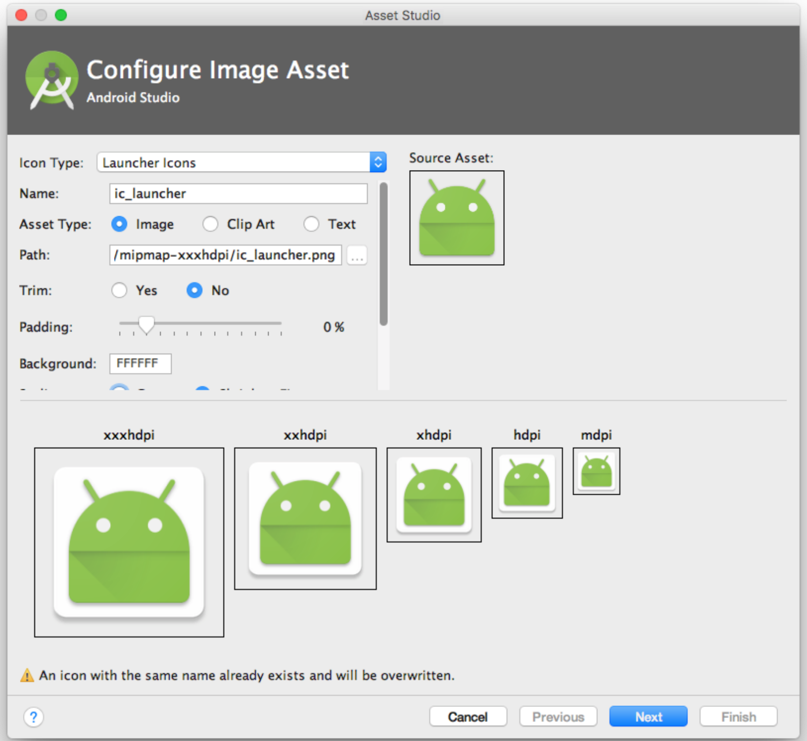
- **Active**: The activity is visible to the user and running.
- **Foreground**: The activity is not visible but still loaded to memory.
- **Destroyed**: The activity has been destroyed and is not accessible.

***Note**: The main activity of a project is specified in the **Manifest file**.*

Changing the Application Icon

Step 1: Creating the Art Assets

1. Right click on the project in Project Explorer.
2. select ***<u>New > Image Asset</u>***
3. Select “**Launcher Icons**” in the Type Dropdown.
4. Select ”Image” as an Asset Type and Select the Desired Image.
5. Press **”Next”** and then **”Finish”**.



Icon Type: Launcher Icons

Name: ic_launcher

Asset Type: [✓ Image] Clip Art Text

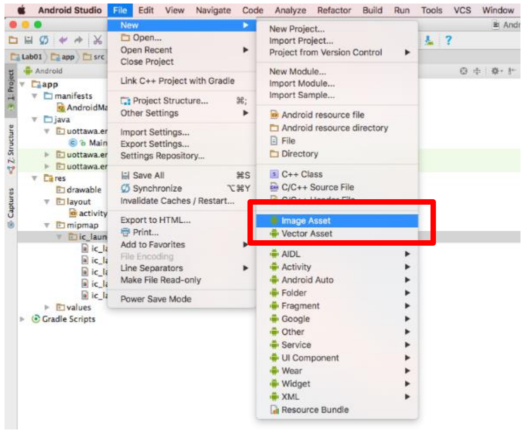
Path: /mipmap-xxxhdpi/ic_launcher.png

Trim: Yes [✓ No]

Padding: 0 %

Background: FFFFFF

An icon with the same name already exists and will be overwritten.



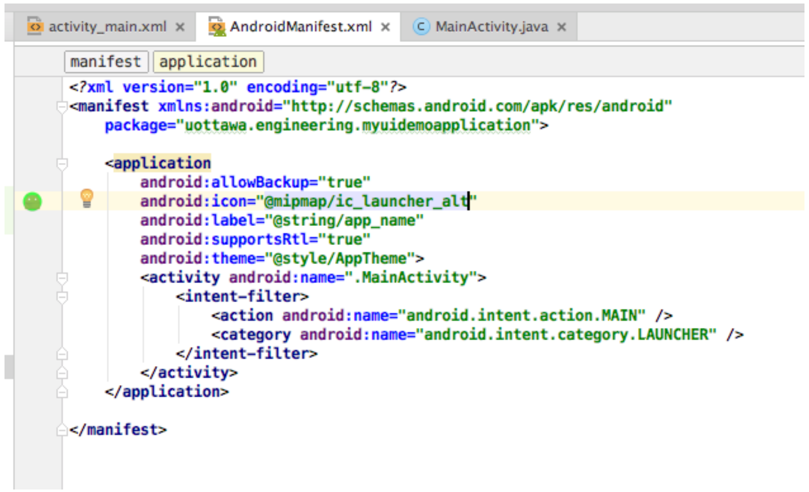
Changing the Application Icon

If you changed the name of the Application Icon You created, you have to assign it in the project Manifest.

Step 2: Updating the Manifest

1. Open the Manifest file (**AndroidManifest.xml)**.
2. Update the Icon Property (**android:icon)** inside the **<Application>** element to another art asset. A miniature preview of your icon will be displayed next to it.

E.g.:
android:icon="@mipmap/ic_launcher_alt"



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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher_alt"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Questions?