Gradle- the basics

Gradle is a build system, like make, it can be very complex. Fortunately Android Studio (AS) handles most of the gradle tasks for you with some exceptions. For instance changing what APIs your device targets, or if you add new components that require adding their gradle dependencies. These often come up when importing someone elses project that uses different SDKs (like when you import my projects), or when upgrading projects. This is a short guide on dealing with those situations.

Open Project

build.gradle (for the application)

first lib or application (see first line in gradle file)

- lib means your building a library
- app means your building an application

```
apply plugin: 'com.android.library' or apply plugin: 'com.android.application'
```

Check your `compileSdkVersion`

`in general use the latest you have installed,check SDK manager occasionally to see if there are updates`

is the version installed ?(red squiggles if not) If not AS will ask you if you want to install it, careful! They are >1GB. or change to one that you have (see SDK Manager)

defaultconfig

project packagename

which versions you support min to target and all in between

buildtypes (not really relevant in this class)

used to support different project flavors for instance a freemium verses paid

proguard is security and obfuscation

dependencies

libraries you need (if possible build with jetpack, AndroidX support which makes all the V4 or V7 appcompatlibrary issues go away. Refactor→Migrate to AndroidX (bit glitchy) or when creating project on the Configure your project screen, select 'Use AndroidX Artifacts')

(see AndroidX Overview
https://developer.android.com/jetpack/androidx

and Migrating to AndroidX
see https://developer.android.com/jetpack/androidx/migrate
)

`gradle.properties`

```
Make sure the following lines are NOT commented out if using AndroidX
android.useAndroidX=true
# Automatically convert third-party libraries to use AndroidX
android.enableJetifier=true
```

`settings.gradle` – whats in this project (project(s), libraries)

Once this is all set make sure you use the appropriate packages in your java files and in your XML files (it will not compile if you do not)

Want to add library as a dependency to an application?

- Create library (see https://developer.android.com/studio/projects/android-library.html)
- Add to project if you wish (see settings.gradle above)

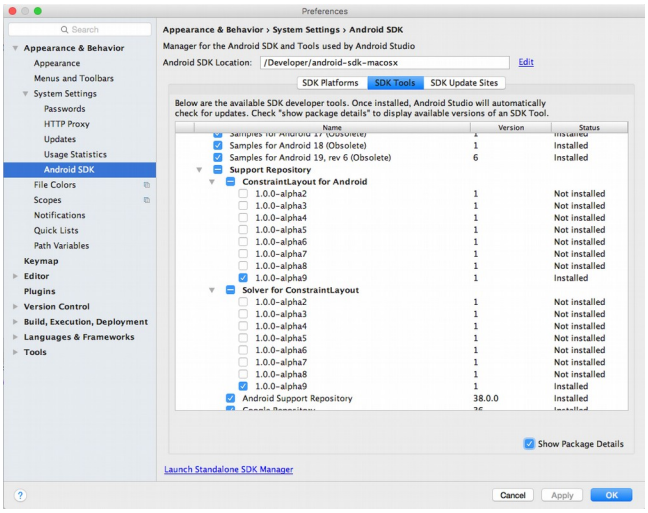
- Add library as a build dependency, in the com.android.application gradle file add a dependency in the dependencies section, like so

```
dependencies {
    implementation  fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:25.1.0'
    implementation project(":bitmap_utilities")
}
```

ConstraintLayout

A lot of you are going to add a constraint layout to existing projects. No worries just see what version you have installed and note that version in your apps build.gradle file. (from stack overflow)

In my case, that support libraries for ConstraintLayout were installed, but I was adding the incorrect version of ConstraintLayout Library in my build.gradle file. In order to see what version have you installed, go to Preferences > Appearance & Behavior > System Settings > Android and move to SDK Tools tab. Check Show Package Details and take note of the version.



Finally you can add the correct version in the build.gradle file

```
dependencies {
        compile fileTree(dir:  'libs',  include:  ['*.jar'])
        compile 'com.android.support.constraint:constraint-layout:1.0.0-alpha9'
        testCompile 'junit:junit:4.12'
}
```

from Stack Overflow:

`compileSdkVersion` is the API version of Android that you compile against.

`buildToolsVersion` is the version of the compilers (aapt, dx, renderscript compiler, etc...) that you want to use.
For each API level (starting with 18), there is a matching .0.0 version.
At IO 2014, we release API 20 and build-tools 20.0.0 to go with it.
Between Android releases we will release updates of the compilers, and so we'll release version .0.1, .0.2, etc...
Because we don't want to silently update these version under you, it's up to you to move to the new version when it's convenient for you. (KP use SDK Manager)
You can use a higher version of the build-tools than your compileSdkVersion, in order to pick up new/better compiler while not changing what you build your app against.