SENG440
Week 1

Mobility
Introduction to Android and Kotlin

Ben Adams, benjamin.adams@canterbury.ac.nz

Mobile-first software design

- What does this mean?
- How does it differ from software designed for desktop computing?

Mobile-first software design

- Ubiquitous
- Situated
- Connected
- Finite

Mobile is a **platform** (Marc Andreeson)

> *A "platform" is a system that can be programmed and therefore customized by outside developers—users—and in that way, adapted to countless needs and niches that the platform's original developers could not have possibly contemplated, much less had time to accommodate.*

Cultural trends in mobile software

- Apps are casual
- Apps are paranoid
- Apps are personal
- Apps are social

Choosing a platform

What is Android?

- A **software stack** for mobile devices that includes
  - An operating system
  - Middleware
  - Key Applications
- Uses Linux to provide **core system services**
  - Security
  - Memory management
  - Process management
  - Power management
  - Hardware drivers


Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Now called ART
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Android **Versioning**

- Over 30 versions in 12 years
- Slowing down, current pace is one large, major release a year
- Android releases have a code name (up to version 9), version number, and API level
- Most recent:
  - Version 14, API level 34
- https://en.wikipedia.org/wiki/Android_version_history

android

Android development tools

Setup Development Environment

- Install Android Studio
  - includes API level 34

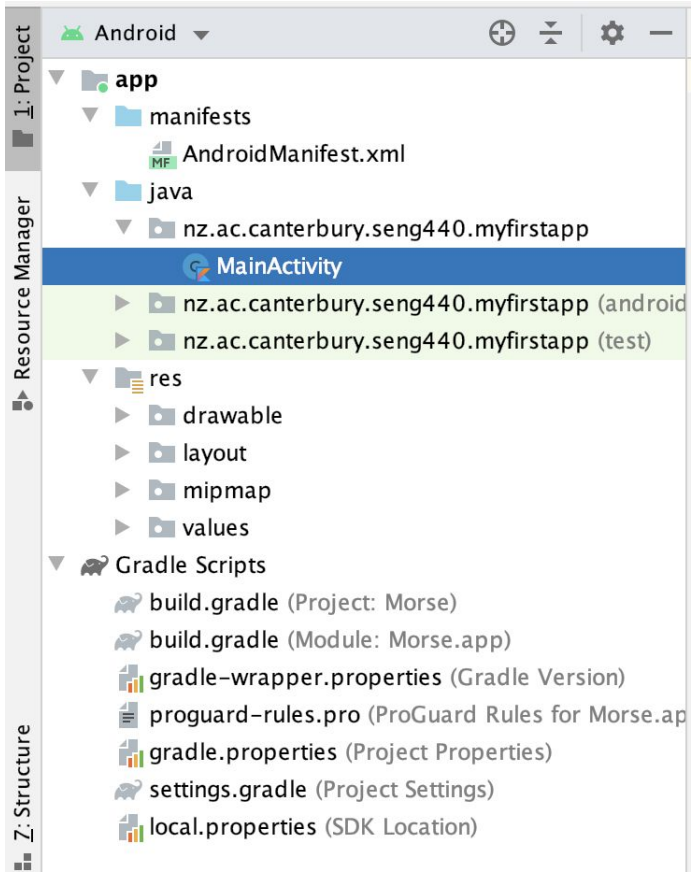
- Use SDK manager to download lower API levels

Elements of Android Projects

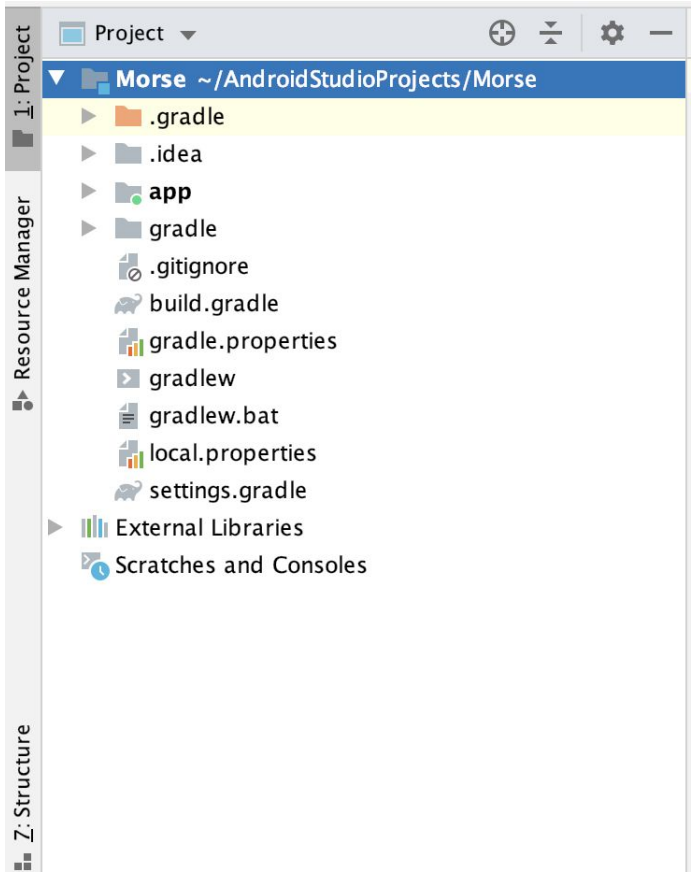
- ***Application Name***
  - Seen by users on app chooser, app list, store
- ***Project Name***
  - Can be different in IDE, often directory
- ***Package Name***
  - Java package name, not using default package
- ***Minimum SDK Level***
  - How far back do you support, 34 levels as of now
  - https://developer.android.com/studio/releases/platforms
- ***Target SDK Level***
  - Device / API you had in mind for app, most recent?
- ***Theme***
  - Look and feel of app, color scheme, various built in themes such as Theme, Holo, Material (Design)

Android Projects

- Creating a project results in multiple files and resources being created



Android Project View

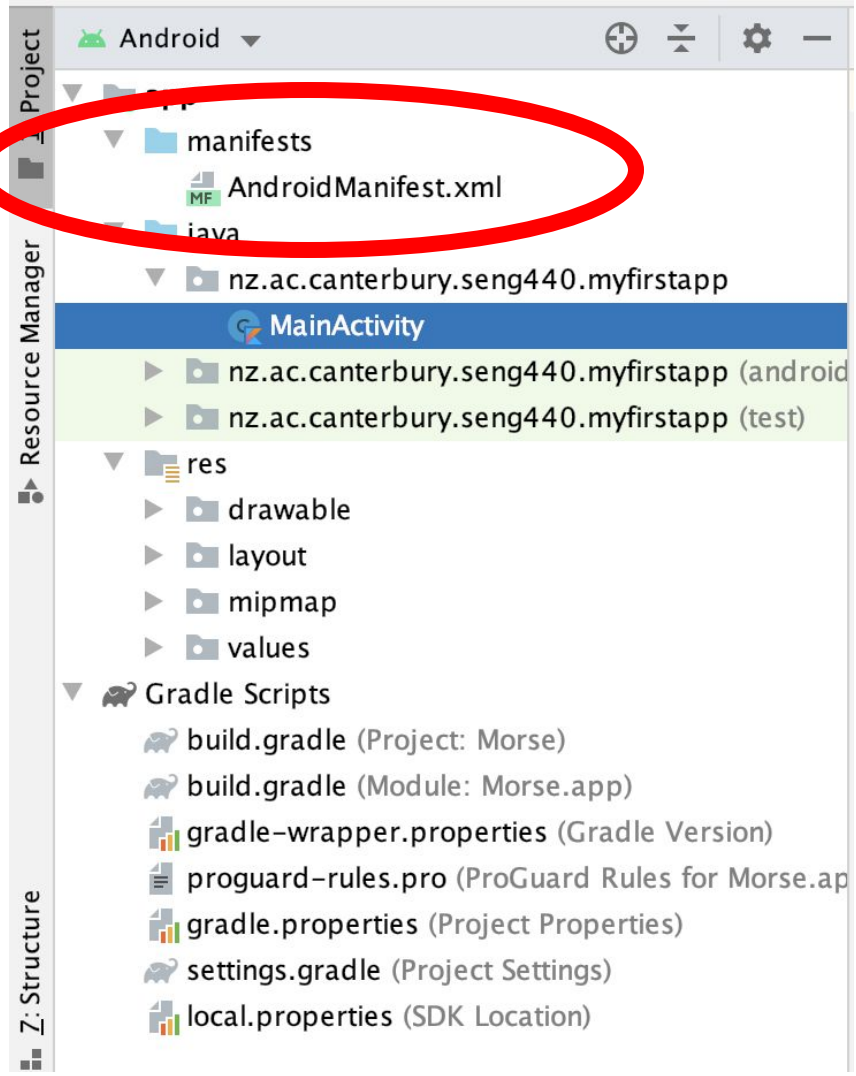


Classic Project View

Android Project Components

Project **Manifest**

- **AndroidManifest.xml**
- Table of contents for your app
- Main **activity**
- Target and min **SDK**
- Declare all the **parts** of your apps:
  - Activities, services
- Request **permissions**
  - Network, location, ...



Android Manifest - Sample

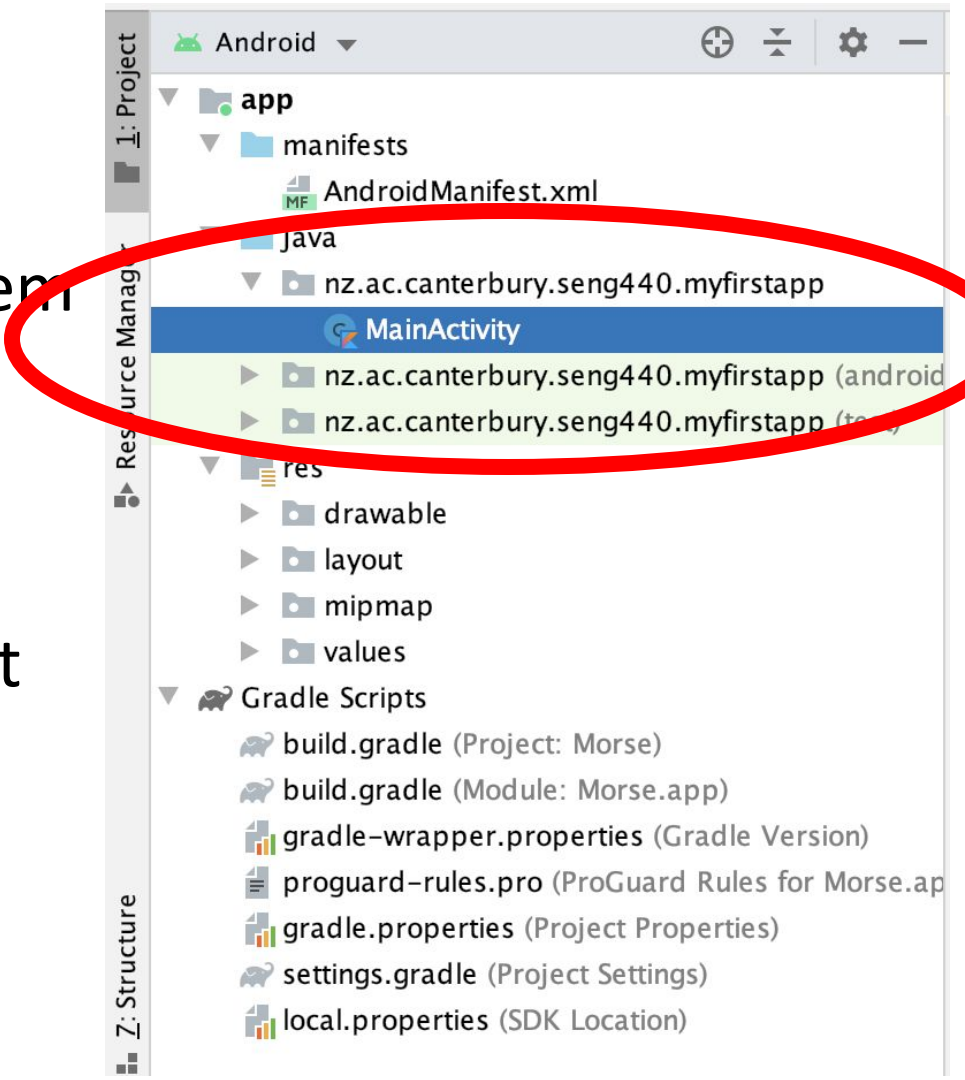
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="nz.ac.canterbury.seng440.myfirstapp">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Morse"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Morse">
        <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>
```

defines Android namespace

defines starting point for App

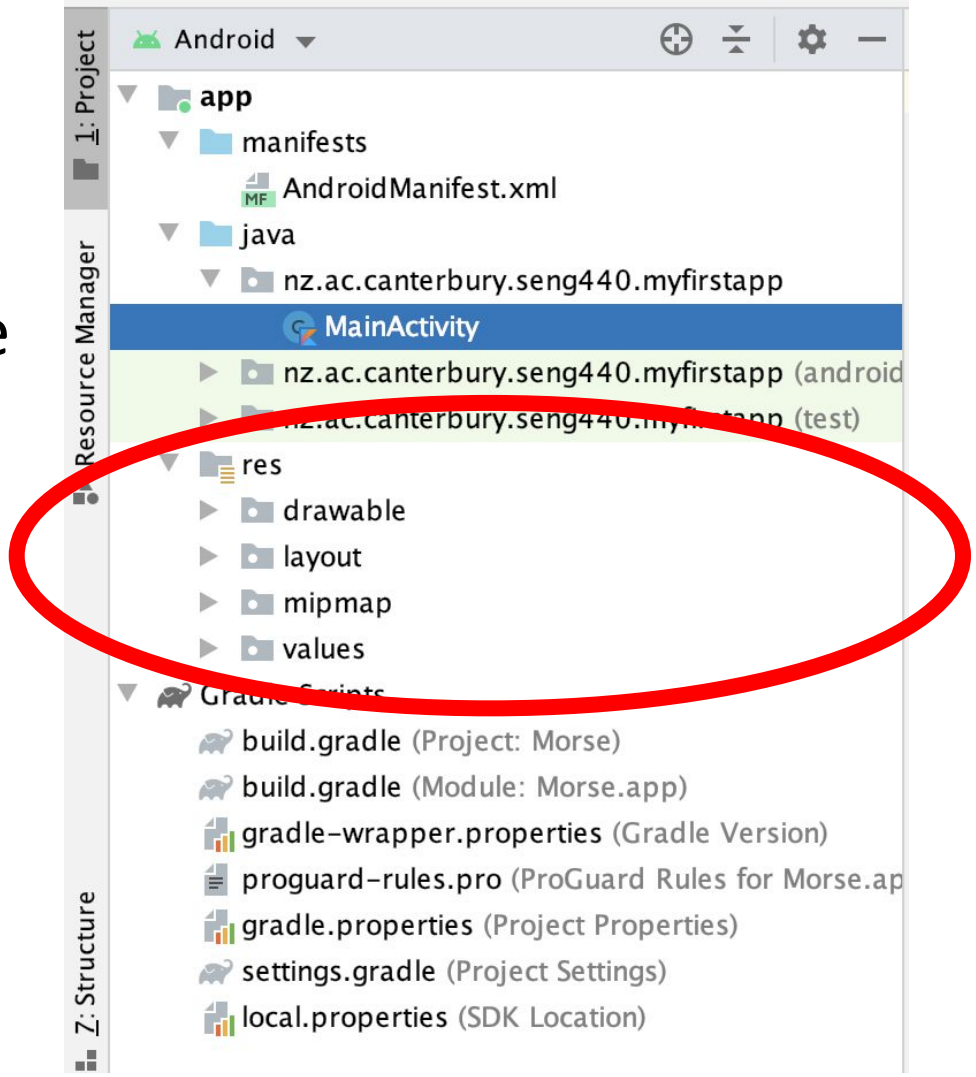
Source Code

- In `java` directory in Android Project View
- Actually in `src` directory on system
- `androidTest` for unit tests involving Android system (req. emulator)
- `test` for non-Android dependent tests



Resources

- Resources in the `res` directory
- Non-source code resources for the app
- Packaged up with app
- Important role and use in development of app



Resource Directories

- **res/drawable** for graphic images such as png, jpeg
- **res/layout** for xml files that define the layout of user interfaces inside the app
- **res/menu** for xml based menu specifications
- **res/values** for lists of strings, dimensions, colors, lists of data
- **res/raw** for other kinds of files such as audio clips, video clips, csv files, raw text
- **res/xml** for other general purpose xml files

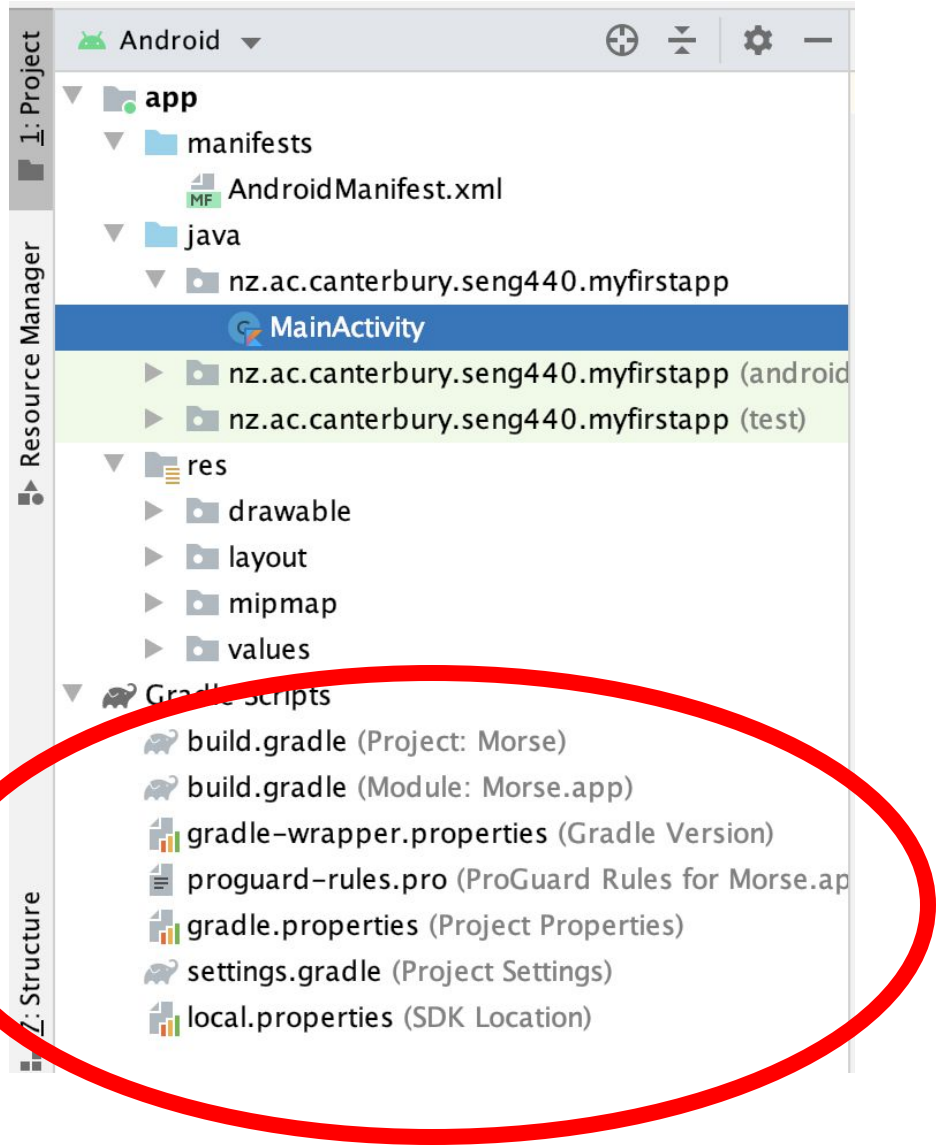
Gradle

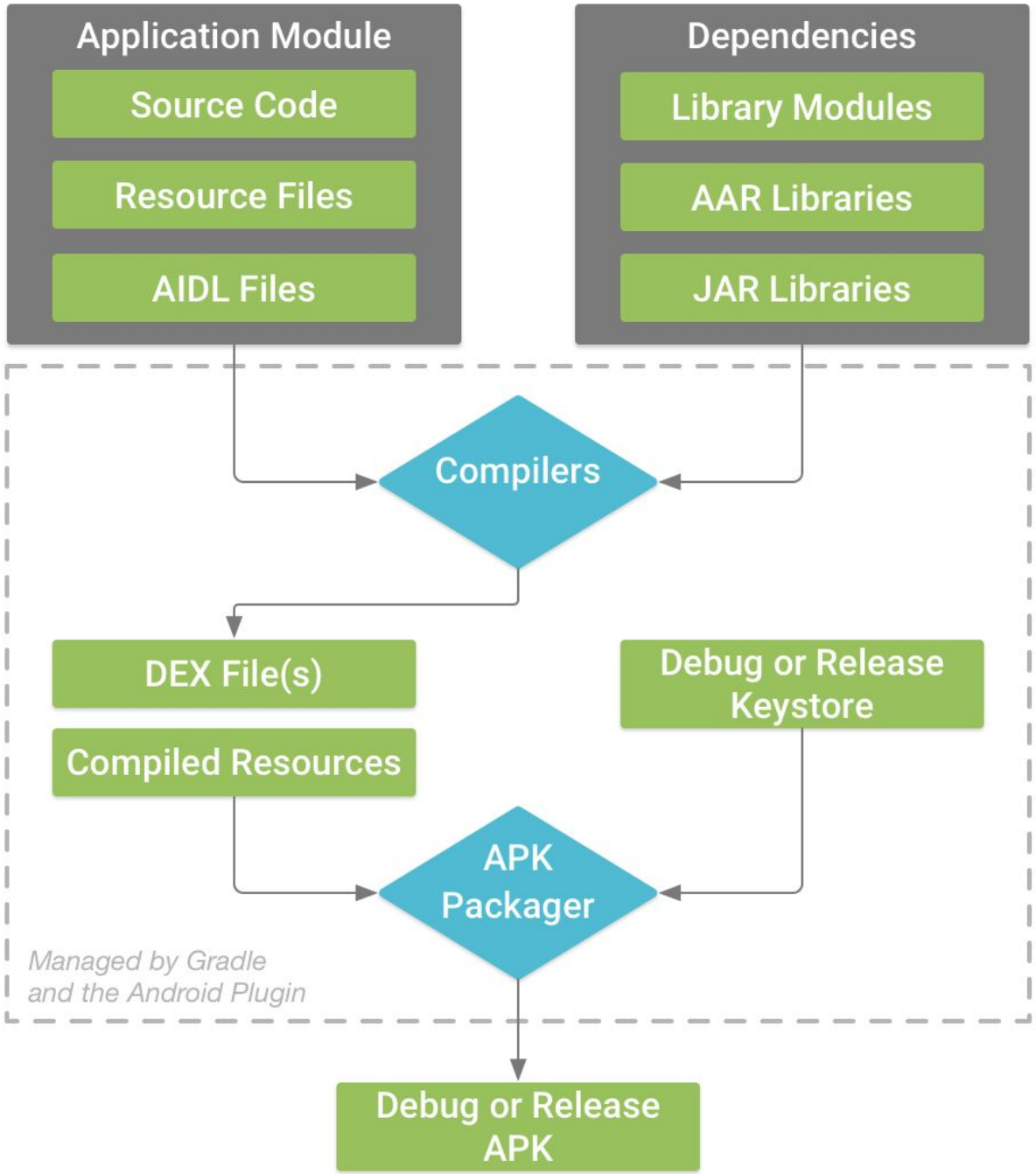
- **.apk** files, **Android Package Kit**
  - Android executables
- Development environment packages the following together:
  - Source code
  - Manifest
  - Libraries
  - Resources

https://developer.android.com/studio/build

Gradle

- Gradle is the build engine that Android Studio uses to convert your project into an APK
- What needs to be created and how to do it
- Like
  - `make` for C/C++
  - Ant/Maven for Java
- `build.gradle` files




Application Module
Source Code
Resource Files
AIDL Files
Dependencies
Library Modules
AAR Libraries
JAR Libraries
Compilers
DEX File(s)
Compiled Resources
Debug or Release Keystore
APK Packager
Managed by Gradle and the Android Plugin
Debug or Release APK

Project build.gradle file

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.
buildscript {
    ext.kotlin_version = "1.3.72"
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath "com.android.tools.build:gradle:4.1.2"
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Module/App build.gradle file

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
}

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.0"

    defaultConfig {
        applicationId "nz.ac.canterbury.seng440.myfirstapp"
        minSdkVersion 27
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
}
```

Android Runtime (ART)
(Dalvik VM before Android 5.0)

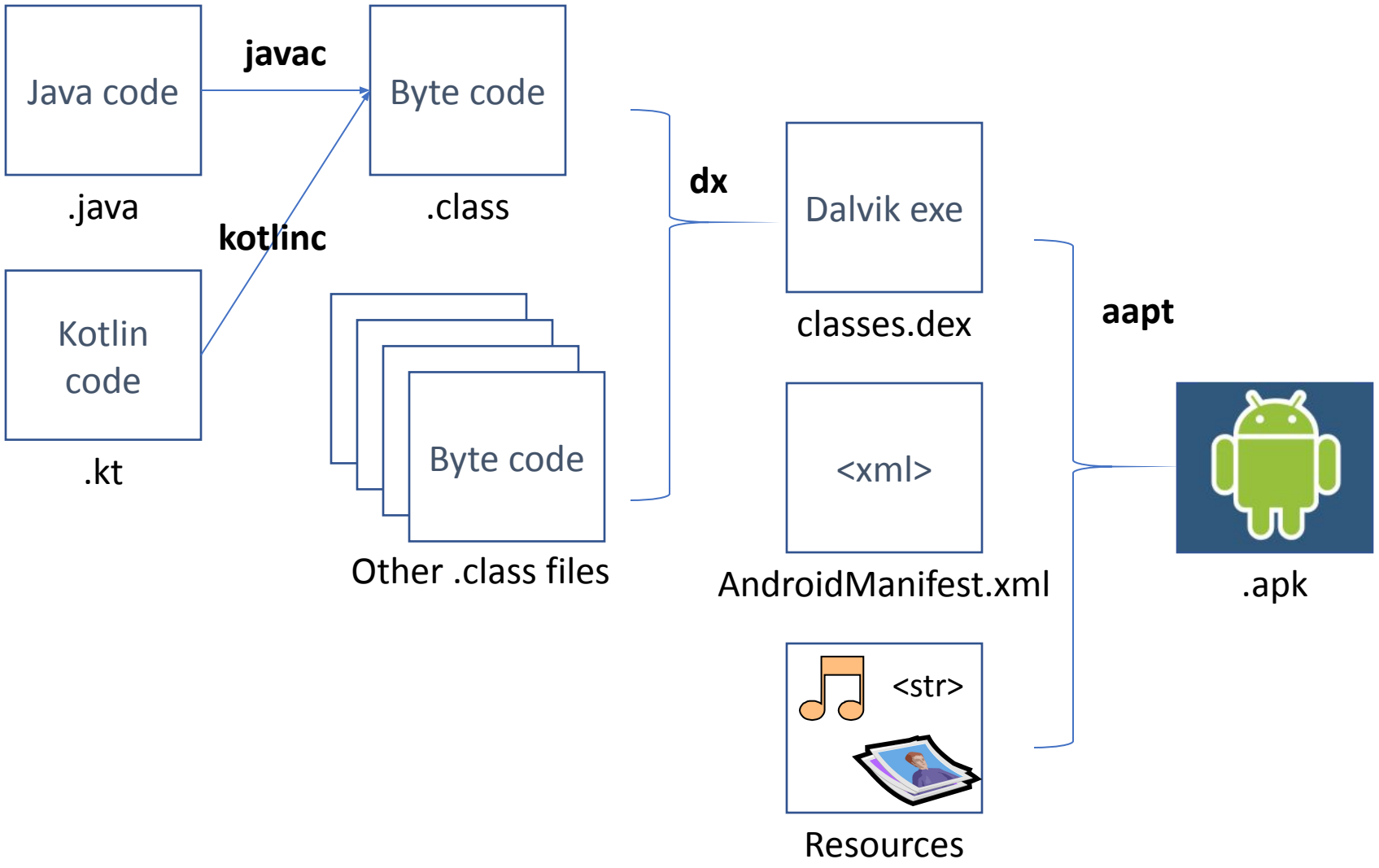
- **Subset of Java** developed by Google
  - https://source.android.com/docs/core/runtime
- **Optimized** for mobile devices
  - Better memory management / battery utilization
  - Ahead-of-time compilation (AOT) – new with ART, Dalvik was JIT
- **Runs .dex** files (bytecode) that are compiled from .class files
- **Creates** compiled Extended and Linkable Format (ELF) **executables** *(**not** JVM bytecode)*
- Introduces new **libraries**
- Does not support some Java libraries like AWT, Swing

http://developer.android.com/reference/packages.html

Applications are **boxed**

- Each app is run in its **own Linux process**
  - Process started when app's code needs to be executed
  - Threads can be started to handle time-consuming operations
- Applications are compiled to **native machine code** with AOT compilation
- Each app is assigned **unique Linux ID**
  - Permissions are set so app's files are only visible to that app

Producing an Android App



Android Debug Bridge (adb)

- Part of **SDK**
- **Command line tool** to communicate with an emulator or connected Android device
- **Check devices** attached / running
- **Install APKs**: e.g., can install packages from places besides Google Play (security?)

https://developer.android.com/studio/command-line/adb.html

Kotlin

- In 2017 Google declared Kotlin an **official language** for Android dev
- Kotlin imposes **less syntactic burden** than Java
- Kotlin promotes a **more functional style** of programming
  - Immutability
  - Higher-order functions
  - Type inference

Main functions

```
fun main(args: Array<String>) {
  println("Goodbye, Pluto!")
}
```

Compiling

```
kotlinc main.kt
kotlin MainKt
```

```
@file:JvmName("Main")

fun main(args: Array<String>) {
  println("Goodbye, Pluto!")
}
```

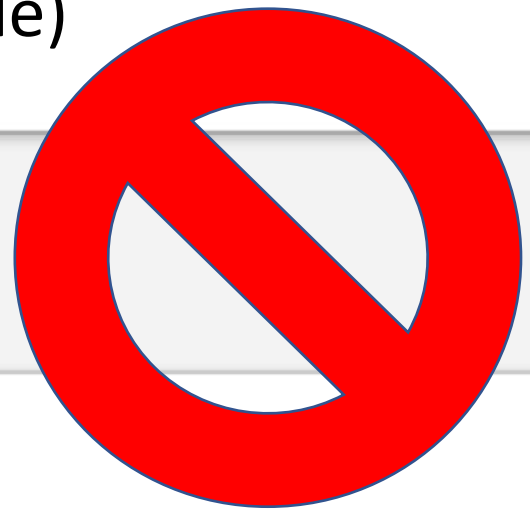
Variables (`var` and `val`)

- Reverse order compared to Java

```
var abbreviation: String = "ChCh"
abbreviation = abbreviation.toUpperCase()
```

- `val` works like Java's `final` (following will not compile)

```
val abbreviation: String = "ChCh"
abbreviation = abbreviation.toUpperCase()
```



Kotlin types

`Double`, `Float`, `Long`, `Int`, `Short`, `Byte`, `Char`, and `Boolean`

- Types can be inferred from context:

```
var abbreviation = "ChCh"
```

- Type conversions must be explicit

```
val x : Int = 64
val y : Long = x.toLong()
val z : Byte = x.toByte()
```

Kotlin floats

```
val a : Float = 3f
```

String templates

- Uses ${} notation for templating

```
println("You passed ${args.size} command-line arguments.
```

Arrays and lists

- Sequential collection structures, including

`Array`, `List`, and `MutableList`

  - `Array`: fixed in size and mutable
  - `List`: fixed in size and immutable
  - `MutableList`: not fixed in size and mutable

```
import kotlin.random.Random

fun main(args: Array<String>) {
  val names = arrayOf("A", "B", "C")
  val winner = names[Random.nextInt(names.size)]
  println("The winner is $winner.")
}
```

Conditional statements

```
val direction = if (Random.nextBoolean()) {
  "L"
} else {
  "R"
}
println(direction)
```

```
println(if (Random.nextBoolean()) "L" else "R")
```

Conditional statements with multiple statements

```
var nLefts = 0
var nRights = 0

for (value in 1..100) {
  val direction = if (Random.nextBoolean()) {
    ++nLefts
    "L"
  } else {
    ++nRights
    "R"
  }
  println(direction)
}

println(nLefts)
println(nRights)
```

when statements

```
val zodiac = when (year % 12) {
    0 -> "Monkey"
    1 -> "Rooster"
    2 -> "Dog"
    3 -> "Pig"
    else -> "Quokka"
}
```

when statements

```
val nDays = when (month) {
  1, 3, 5, 7, 8, 10, 12, -> 31
  2 -> 28
  else -> 30
}
```

```
val generation = when (year) {
  in 1981..1996 -> "Millenial"
  in 1946..1964 -> "Baby Boomer"
  else -> "?"
}
```

when statements

```
val quadrant = when {
   x > 0 && y > 0 -> "I"
   x < 0 && y > 0 -> "II"
   x < 0 && y < 0 -> "III"
   x > 0 && y < 0 -> "IV"
   else -> "None"
}
```

Function form

```
fun name(param1: Type, param2: Type, ...): ReturnType {
  // body
}
```

```
fun roll2d6() = Random.nextInt(6) + Random.nextInt(6) +
```

Classes

```
class Model(param1: Type, ...) {
  val prop1: Float = ... // read-only property
  var prop2: String = ... // read-write property

  fun method(...) {
    ...
  }
}
```

Constructors in Kotlin

```
class Point(x: Float, y: Float) {
  var x: Float = x
  var y: Float = y
}
```

```
class Point(var x: Float, var y: Float) {
}
```

Using classes

```
class Point(var x: Float, var y: Float) {
  override fun toString = "($x, $y)"
}
```

```
fun main(args: Array<String>) {
  val p = Point(3, 4)
  println(p)
}
```

Class getters and setters

```
class Person {
    var nameChanges: Int
    var name: String
      set(value) {
          nameChanges++
          field = value
      }
    constructor(name: String) {
        this.name = name
        this.nameChanges = 0
    }
}
```

Lambdas

- Return type is the last statement in the body

```
{ param1: Type, param2: Type2 ->
  body
}
```

```
val xs = intArrayOf(1, 2, 3, 4, 5, 6, 7)
xs.forEach({ x -> println(x) })
```

```
xs.forEach() { x -> println(x) }
```

```
xs.forEach { x -> println(x) }
```

TODO list

- Join class Slack page. I will send out invites soon.
- Read the syllabus
- Read the description of Project 1
- Install Android Studio - https://developer.android.com/studio/install
- Familiarise yourself with Kotlin:
  - Read Basic Syntax , Basic Types, Control Flow, Functions. Write a `main` function that declares some variables, executes some computation of your choice using a helper function or a lambda, and uses some additional feature of Kotlin you have discovered. Test your code in the Kotlin Playground or in the Kotlin REPL in Android Studio.
- Take a look at the Week 2 tutorial notes. I will go through these in lecture next week, but the more you read in advance the better.

How to access developer mode?

[How to Enable Developer Options and USB Debugging on Android]