Android Mobile Application Development
Day 1: Course Overview and Introductions

Class Introductions
Introduce myself (slides)
Introduce the class
Canvas https://canvas.cmu.edu/courses/31825

Go over Syllabus

Class list
- Share one fun/unique thing about yourself

Mobile platform landscape (slides)
- Over 3.5 million apps in the Google Play store, and over 2 million in Apple's app store

Mobile App Development
- Native apps
  - Developed in the device's native language
  - Use native tools
  - Access to native APIs
  - Native look and feel
  - Fastest performance
  - Distribute through the app store
  - Internet access not usually required
  - Can't run on multiple platforms
  - Multiple code bases for multiple platforms
- Native app frameworks
  - Web stack
    - Phonegap and Cordova
  - Native stack
    - React Native (Facebook React)
    - Ionic (Google Angular 2)
    - Xamarin (Microsoft C#)
    - Flutter (Google Dart)
    - Single code base
    - Runs on multiple platforms
    - Good performance, closest to native
    - Doesn't always look and feel native
    - Dependent on the components available
    - Depending on functionality will get you 60-80% there, the rest native
    - Still need some knowledge of native app development
- HTML 5 web app
  - Built with HTML, CSS, and JavaScript or Web framework
  - Takes advantage of web development skills
  - Runs on the Internet through a browser
  - Internet access required
  - Single code base
  - Can run on multiple platforms

- o Updated instantly
- o Many frameworks available to help with development (interfaces, navigation, etc)
- o Distributed through web sites(unrestricted), not app store
- o No/limited access to native APIs limits functionality (ie camera access)
- o UI can feel not native
- o Performance can be an issue
- Hybrid apps
  - o Takes HTML5 web apps and wraps them in a native package (ie Cordova)
  - o Can take advantage of more native APIs (if plugins are available)
  - o Distributed through app store
  - o Takes advantage of web development skills
  - o Runs through an embedded browser
  - o Single code base
  - o Can run on multiple platforms
  - o Multiple frameworks, plug-ins, and wrappers to coordinate
  - o Need to understand native platform to develop or customize plug-ins
  - o UI can feel not native
  - o Performance can be an issue
  - o Still the wild west of development

Getting Started w/Android
The Android operating system was originally developed by Android Inc which Google purchased in 2005.
Android 1.0, the first commercial version, was released on 9/23/08.
It now has 72% of the worldwide market share compared with 28% for iOS. (slide)

Android Development
Developer web site http://developer.android.com
Android Studio (or Download link)
http://developer.android.com/studio/ (scroll down for system requirements)
- Android development can be done on Windows, Mac OS, or Linux systems
- Android Studio
  - – Integrated Development Environment (IDE)
  - – Android Software Developer's Kit (SDK)
  - – Emulator to run, test, and debug your apps on different virtual devices
  - – Tools and profilers for testing
  - – Initially released in 2013, previously the SDK was bundled with Eclipse
  - – Can also use the SDK with the command line or another SDK
- Java programming language
  - – Although Android uses Java it does not run .class or .jar files, it uses its own format for compiled code
- Kotlin programming language
  - – Kotlin was developed by JetBrains in July 2011
  - – In February 2012 Kotlin was open sourced
  - – Kotlin 1.0 was released in February 2016
  - – Google announced Kotlin support at Google IO in 5/17
  - – Integrated into Android Studio 3.0
  - – Kotlin-first https://developer.android.com/kotlin/first

- In May 2019 Google announced that Kotlin was now its preferred language for Android development.
  – Kotlin is cross platform and fully interoperates with Java

Use the SDK Manager (Tools | SDK Manager) to download additional tools and components.

Android Setup
The first time you launch Android Studio it will take you to a setup wizard.
Configure | SDK Manager to install other needed SDK components.
OR
Tools | Android | SDK Manager to install other needed SDK components.
SDK Platforms (show package details)
- Android Tiramisu (API 33)
- Show Package Details
  o Android SDK Platform
  o Sources for Android
  o Google APIs Intel x86 Atom System Image (for the emulator)
SDK Tools
- Android SDK Build tools
- Android Emulator
- Android SDK Platform-Tools
- Android SDK Tools
- Documentation (do this later)
- Intel x86 Emulator Accelerator (HAXM Installer) (do this later)
- Layout Inspector image server for API 29-30 (do this later)

SDK Update Sites
All should be checked.

Chose top license, click Accept. Install.

Android EcoSystem
- Google -> phone manufacturers (OEMs)
- OEMS -> carriers
- Android updates take about 3-6 months for a new version is available on a carrier
- Android versions – version, dessert name, API
  – When creating a new project you need to decide what Android versions to support

Android Devices
Android runs on devices that are all different
- Screen sizes
- Processor speed
- Screen density
- The number of simultaneous touches the touch screen can register
- The quantity and positioning of front and back cameras
- Bluetooth

Screen densities https://developer.android.com/guide/practices/screens_support.html

Pixel densities https://developer.android.com/training/multiscreen/screendensities
The density-independent pixel, dp, is an abstract unit based on the physical pixel density of the screen.
1 dp is equal to 1 physical pixel on a medium density screen (160 dpi screen).
At runtime, Android automatically handles the coversion of dp units based on the density of the screen.
The conversion of dp units to screen pixels is: px = dp * (dpi / 160).
So on a 480 dpi screen, 1 dp equals 3 physical pixels.

Android Studio
Let's create our first Android app and make sure our environment is all set up.
From the Welcome screen chose Start a new Android Studio Project (File | New | New Project)
Select a Project Template: In the Phone and Tablet tab pick Empty Compose Activity.
Name: HelloAndroid
Package name: the fully qualified name for the project
Save location: the directory for your project (make sure there is a directory with the name of the project after the location)
Language: Kotlin
Minimum SDK: API 21: Android 5.0 Lollipop (21 is the minimum API for Jetpack Compose)
(Help me choose will show you the cumulative distribution of the API levels)
Leave legacy libraries unchecked (using these restricts usage of some newer capabilities)
Finish

(Preferences | Editor | Color Scheme | General to change scheme to presentation)

Android Studio Tour
https://developer.android.com/studio/intro/
The left most pane is the Project Tool Window which provides a hierarchical overview of the project file structure.
The default setting is the Android view which is what I usually leave it in but there are other choices for viewing your project.
Above that is the navigation bar which provides another way to navigate your project files.
Also notice the tabs on the left which will change what is shown in in the left pane.

The middle pane is the editor window which shows the file you're editing. Above the editor on the right are controls for the view mode.
If you have more than one file open you will see tabs right above the editor.
Above that is a toolbar for compiling and running your project.

The tabs on the right which will change what is shown in in the right pane.

The bottom window shows the status. There are different tabs at the bottom to choose what you want to see and you can also hide it.

Android Virtual Device (AVD)
https://developer.android.com/studio/run/managing-avds

Android Studio lets you define Android Virtual Devices (AVD) so you can run your app in emulator software on your computer. Emulators imitate the software AND hardware environments of the actual device you define as an AVD. An AVD is a configuration that defines a hardware profile, system image, storage area, and other properties of a specific Android device.

Tools | Device Manager (or Device Manager ⬚ in the toolbar)
Create a virtual device
Category: Phone
Chose a phone like the Pixel - size: large, density: xxhdpi or the Nexus 4 has the size: normal, density: xhdpi
System Image: API 33 Tiramisu or S API 31 Android 12 (download the needed system image)
AVD Name: Pixel 3 API 33 (or something else descriptive)
Leave the rest as is and Finish
Now you should see your AVD as a virtual device.
Close the AVD Manager.

By default when you run the emulator it runs in a tool window in the right pane (Emulator tab on the bottom right). You can also run the Android Emulator as a separate application.

Setup:
File > Settings > Tools > Emulator (or Android Studio > Preferences > Tools > Emulator on macOS), unselect Launch in a tool window and click OK to run the emulator standalone.
If the Emulator window doesn't automatically appear, open it by clicking View > Tool Windows > Emulator.

Project Files
Now let's take a closer look at what was created in our project. By default, Android Studio displays your project files in the **Android** view. This view does not reflect the actual file hierarchy on disk, it is simplified for ease of use.
In the Android view you see 3 main areas of your app
1. manifests: The manifest file describes the fundamental characteristics of the app and defines each of its components. Every part of your app will be declared in this file. It acts as a blueprint for putting all the pieces of an app together.
    a. Every application must have an AndroidManifest.xml file (with precisely that name) in its root directory.
    b. The package name is the unique identifier for your app on devices and the Play store
    c. The application tag describes various characteristics of your app
    d. Our application also has an activity element with the name MainActivity
        i. the action MAIN which tells Android which activity should be run when the app starts
            1. Equivalent to a main method used in other languages
        ii. The category LAUNCHER tells Android that this activity can be launched from the home screen
2. java: These are your Kotlin (or Java) files that contain all of the core logic for your app. It is named java for historical reasons but also highlights the fact that Kotlin and Java can work interchangeably in Android apps.
    a. uses the package name
    b. Unless you're doing testing, you always want to look in the first one (that doesn't say test)
    c. MainActivity.kt defines a class for us called MainActivity that extends ComponentActivity.
        i. An activity is a single module of application functionality that often correlates to a single screen

1. Usually associated with one screen
2. Written in Java or Kotlin
   ii. ComponentActivity is the base class in Android for activities that use Jetpack Compose.
   iii. The : in Kotlin creates a subclass from a superclass
   d. ui.theme is a package that contains files creating the theme and other design elements
   e. Ctrl-Q (PC), Ctrl-J (Mac) opens the documentation for wherever your cursor is (bottom right tab).
3. res: these are static resource files that include
   a. Drawable – images for the project
   b. Mipmap – includes files such as launcher images (app icons are called launcher icons)
   c. Values – resource files that include values for strings, styles, colors, and other resources
4. Gradle: Gradle is the build tool in Android Studio. Gradle controls the following:
   a. compiles code, dependencies and resources to executable code
      i. builds the app package file known as an APK file
      ii. APK is the executable format for distributing Android apps
   b. Declaring dependent code such as libraries (dependency management)
   c. Determine what devices the app is built too run on
   d. App signing for the Google Play store
   e. Running automated tests

If you switch to the Project view you'll see the actual file structure of the project. There are a lot of other files, but we're not going to worry about most of them.

App Preview
To the right and above the right pane are controls for the view mode
- Design shows the layout preview
- Code shows the Kotlin code
- Split shows both

In MainActivity a preview has been defined using the @Preview annotation.
If you build your project you can see the preview.
Go into Split mode and click on build and refresh or Build | Make Project
Previews let you quickly see the layout of your activity without having to compile and run the app.
Every time you change your layout you will need to build and refresh.
If you change the string passed to the Greeting() function in the DefaultPreview() function you will see this reflected in the preview.
More significant changes will require a build and refresh before being reflected in the preview.
Previews are not interactive, for that you must run the app.

Run the App
https://developer.android.com/studio/run
To run your app and be able to interact with it you must use an AVD in the emulator or run the app on a device.
In the toolbar bar you should see the AVD you just sent up and a play/run button next to it.
Click the Play button to build and run your app.
You might have to wait a bit for the Emulator to start and launch your app.

You can view details in Android Studio about the build process by clicking View > Tool Windows > Build (or by clicking Build in the tool window bar at the bottom). The window displays the tasks that Gradle executes in order to build your app,

The first time an emulator starts it can take a few minutes.

You can leave the emulator running so it will be faster for subsequent runs.

You might need to unlock your device – just swipe the padlock icon upwards and you should see the sample app.

Most newer Android devices have 3 virtual buttons.

Left button (back arrow) goes back to the last activity

Middle button (home) takes you to the home screen

Right button shows a list running apps

The favorites tray is at the bottom with the all apps button in the center.

For Thursday
- Finish setting up Android Studio if you haven't already
- Download the documentation when you're on the CMU network