ATLS 4120/5120: Mobile Application Development
Week 8: Android Development Overview

Android
Android 1.0, the first commercial version, was released on 9/23/08.
It now has over 75% of the worldwide market share compared with 20% for iOS. (slide)
In the US iOS has 54% and Android 45%. (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 OSX, or Linux systems
    - Note that MacOS 10.15 Catalina is NOT listed as supported. Please stay on Mojave.
- 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
- Announced Kotlin support at Google IO in 5/17 and was integrated into Android Studio 3.0
- eXtensible Markup Language (XML)

As of Android Studio 2.2, the IDE comes bundled with a custom OpenJDK build which contains the Java Development Kit (JDK) and a bunch of additional fixes to make the IDE work better (such as improved font rendering).

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

Older versions required you download the Java Development Kit (JDK) and Java Runtime Environment (JRE) separately, but this is no longer required.

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 10.0(Q) (API 29)
- Show Package Details
    - Android SDK Platform
    - 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)

SDK Update Sites
All should be checked.

Chose top license, click Accept. Install.

Android Releases

- Platform versions and stats on the Android Developer site dashboard http://developer.android.com/about/dashboards
- Google -> phone manufacturers (OEMs)
- OEMS -> carriers
- Android updates take about 3-6 months for a new version is available on a carrier (slide)
- When creating a new project you need to decide what Android versions to support

Android Devices

Android runs on devices that are all different http://developer.android.com/about/dashboards (scroll down)

- 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
https://developer.android.com/training/multiscreen/screendensities
The density-independent pixel is equivalent to one physical pixel on a medium density screen (160 dpi screen).
At runtime, the system transparently handles any scaling of the dp units, as necessary, based on the actual density of the screen in use.
The conversion of dp units to screen pixels is: px = dp * (dpi / 160).
So on a 240 dpi screen, 1 dp equals 1.5 physical pixels.

Android Terminology

- An activity is a single, defined thing a user can do
  - Usually associated with one screen
  - Written in Java or Kotlin
  - Android Studio has activity templates
- A layout describes the appearance of the screen
  - Design view
  - Written in XML
- The emulator lets you define Android Virtual Devices(AVD)
  - A device configuration that models a specific device
- Simulators imitate the software environment of the device but not the hardware.
  - They have access to all the host(Mac) hardware's resources
- Emulators imitate the software AND hardware environments of the actual device

- A flight simulator simulates a flight. If it was a flight emulator then you'd actually be transported to the destination.

Android Studio
https://developer.android.com/studio/projects/create-project
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)
In the Phone and Tablet tab pick Empty 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: Java
Don't check either Include check boxes
Form factors: Phone and Tablet (leave others unchecked)
Minimum SDK: API 21: Android 5.0 Lollipop (21 is the minimum API for Material Design)
We don't need to support instant apps
Check Use android.* artifacts
Finish

(Preferences | Editor | Color Scheme 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.
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. You'll see different editors here depending on what type of file you're editing. If you have more than one file open you will see tabs right above the editor.
Above that is the navigation bar which provides another way to navigate your project files.

When you're editing a layout's xml file the preview window shows on the right side. You can toggle this using the tabs on the far right or hide it.

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
Tools | AVD Manager
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: Q API 29 Android 10 or Pie API 28 Android 9 (download the needed system image)

AVD Name: Pixel 3 API 289(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.

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: Every application must have an AndroidManifest.xml file (with precisely that name) in its root directory. 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. The package name is the unique identifier for your app on devices and the Play store
    b. The application tag describes various characteristics of your app
    c. Our activity also has a tag with the name MainActivity
2. java: These are your Java files, using the package name
    a. Unless you're doing testing, you always want to look in the first one (that doesn't say test)
    b. MainActivity.java defines a class for us called MainActivity that extends AppCompatActivity. Extend in Java creates a subclass from a superclass. The AppCompatActivity class is used for all Android activities.
    c. Ctrl-Q (PC), Ctrl-J (Mac) opens the documentation for wherever your cursor is. Context sensitive Java and Android documentation can be accessed by clicking on the declaration.
3. res: these are resource files that include
    a. Drawable – images for the project
    b. Layout – the layout files for the user interface in xml
        i. activity_main.xml defines the one TextView that was created for us and its properties including ConstraintLayout properties.
        ii. Two tabs at the bottom let you switch between text and design mode.
    c. Mipmap – includes files such as launcher images (app icons are called launcher icons)
    d. Values – resource files that include values such as the values for strings, styles, colors, etc

The logic is in Java and everything else is a resource.
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.

R class
I do want to point out just one file because you'll see it referred to.
(make sure you build first)
App/build/generated/not_namespaced_r_class_sources/debug/r/com/example/helloandroid/R.java
The R.java file acts as an index to all of the resources identified in your project. This file is automatically generated for every Android project and the "R" stands for resources. Any time you change, add, or remove a resource, the R class is automatically regenerated. We will use the R class a lot but you should **never** manually edit the R.java source files.

Also, you're going to see the term Gradle often. Gradle is the build tool in Android Studio. It compiles and builds the app package file known as an APK file.

Run the App
https://developer.android.com/studio/run
In the toolbar bar you should see the AVD you just sent up and a play 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). 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.
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 show a list running apps (like a double tap of the home button on iOS)
The favorites tray is at the bottom with the all apps button in the center.

Layout
https://developer.android.com/studio/write/layout-editor
Open the activity_main.xml file and go into the Design mode. (tabs at bottom)
Underneath the file tabs you can choose between Design view and/or blueprint view.
Moving across to the right is a button that let's you change the orientation and create layout variants.
Then there's a drop down that let's you chose what device you preview, the API, app theme, and language.
To the right of those you'll see zoom controls.
Below those you'll see a bunch of controls for Autoconnect. For now enable Autoconnect is on by hovering over the magnet looking icon.
Now take a look in the palette to the left of the layout editor to see all the layouts and widgets available for use.
A text view has already been added for us. Click on it in the Component Tree and all the way to the right open up the Attributes pane. Here you can see some of the attributes for the TextView. You can see more by opening different sections include all attributes at the bottom.
Look at the XML for TextView by going into Text mode.
You can see all the same properties including the constraints that were automatically added.

String Resources
Notice the textview has `android:text="Hello World!"`
It's not good practice to hard code string values in your xml file, you should use string resources.
strings.xml in the values folder stores all the string resources.
Open strings.xml and you'll see there's already a string for the name of the app.
Add this resource with some initial value so you can see it worked.
```
<string name="text_message">Hello Android </string>
```
Back in activity_main.xml change the textview to `android:text="@string/text_message"`
The '@' sign means it's defined as a resource.
Now go into Design mode or Preview and you will see our textview has the new string value.
You can click anywhere on `@string/text_message` and click cmd-B (mac) (cntrl B on a pc) to automatically open the strings.xml file where the values for the string resources are stored.

Let's also make the text larger by either scrolling in the Properties pane to textSize or adding in the xml
`android:textSize="48sp"`
We will always use the "sp" unit for font sizes as it allows the font size to scale for the user based on their settings. (scale-independent pixel)
sp has scalable ratio: sp = px * ratio * scale. Where ratio never changes, but scale is user configurable. This will allow the font size to scale such as for people who set larger font sizes in their settings.
If this is a value you're going to be using throughout your layout you should create an entry in the dimens.xml file with a name so you can easily reuse it.

Now let's change the background color through the xml.
In the top level <ConstraintLayout tag before the close tag > add
`android:background="#ff6232"`
Notice that a small colored square appears in the gutter to the left. Click on it and it brings up a color wheel. Open up the preview and you'll see the result.

Click on Run app to run your app in the emulator again (arrow/stop icon). Or stop and then run.

Code completion
The editor in Android Studio has code completion just like in Xcode. As you type you will get selections. To accept the top most suggestion, press the Enter or Tab key on the keyboard. To select a different suggestion, use the arrow keys to move up and down the list, once again using the Enter or Tab key to select the highlighted item.
In the XML file because everything starts with "android" it gets old typing that. You can just start typing what comes after the colon and the code suggestions will still come up. So instead of typing `android:background` just start typing `background` and you'll see `android:background` as a choice.
To see other suggestions, click on a word and then use cntrl-space and the editor will show you a list of alternative suggestions.

Can you guess how you would change the font color for the textview?
In <TextView> add android`:textColor="#FFFFFF0D"`

Note: To open a project in Android Studio there is no one file you can click on to open the project (similar to the xcodeproj file). To open a project go into Android Studio and open an existing project from there. If you're opening a project that you did not create, such as my samples, chose import project instead to avoid configuration errors.