# Introduction to Android and Android Studio

Google Developer Documentation: developer.android.com

Google Tool Documentation: developer.android.com/tools

## Java Development Kit

Install Oracle JDK

OpenJDK might not work as well as Oracle JDK

Android Studio itself can run on Java 7 or 8, although Android devices only support Java 7

Set your JAVA_HOME environment variable to the path of the directory where you installed the JDK

Windows video: https://www.youtube.com/watch?v=GwU4AJn0Txg

## Android Studio Setup

Using Android Studio in open labs (tutorial document)

Tools→Android→SDK Manager (launching stand-alone gives more detail)

SDK/API-levels, Build Tools, Emulator Images

If your code is to compile both on your laptop and on the lab machines, install the same SDK/API-level that's in the labs (Android 5.0.1/API-level 21, Build Tools 23.0.2)

[Open Asteroids starter code]

Running apps on emulator

http://developer.android.com/tools/devices/index.html

http://developer.android.com/tools/devices/emulator.html

Enable hardware virtualization in BIOS settings (Windows)

Tools→Android→AVD Manager (create, run emulators)

Run Asteroids starter code on emulator

Running apps on hardware device

Connecting Your Fire Tablet for Testing (https://developer.amazon.com/public/resources/development-tools/ide-tools/tech-docs/05-setting-up-your-kindle-fire-tablet-for-testing)

Device USB driver (Windows)

Enable development on your device

Other Vendors: Google for driver and any other setup instructions

EX: Samsung: http://developer.samsung.com/technical-doc/view.do?v=T000000117

Google “samsung android usb driver”

Run Asteroids starter code on device

## Android Studio Tour

Open project

Tool windows (show/hide)

Project window

Project – physical file structure, Android – flattened view, easy access to most important files

Build

Gradle overview & console

Build Asteroids starter code

Source code structure

Root Folder: Project.iml (project config file); local.properties (SDK directory); settings.gradle, build.gradle (global Gradle config files); gradlew{.bat} (Gradle command-line script)

Modules (parts of the application: apps, libraries)

“app” Folder: app.iml (module config file); build.gradle (module Gradle config file)

“app/src” Folder: “main” folder (module source code); “test” folder (module local test code); “androidTest” folder (module on-device test code)

“app/src/main” Folder: “java” folder (Java source code); “res” folder (resource files: XML for views, strings, styles, etc., images & icon files); “assets” folder (other data files required by application)

“AndroidManifest.xml” File: Main app configuration file (Permissions, Activities)

Gradle sdk versions override the manifest uses-sdk element (which can safely be deleted)

Build.gradle files

If you change the *.gradle files, you must sync your project with the updated Gradle files

Tools→Android→Sync Project with Gradle Files

Normally, you would make targetSdkVersion and compileSdkVersion the version of the latest SDK you have installed with Android Studio, and buildToolsVersion the latest version of the build tools you have installed with Android Studio.

HOWEVER, IF YOU WANT YOUR CODE TO COMPILE ON BOTH YOUR LAPTOP AND THE OPEN LAB MACHINES, make targetSdkVersion and compileSdkVersion the version of the latest SDK in the open labs [which is Android 5.0.1/API-level 21], and buildToolsVersion the latest version of the build tools the labs have installed with Android Studio [which is Build Tools 23.0.2].

**min sdk version** is the minimum version of the Android Operating System required to run your application.
**target sdk version** is the version of Android that your app was created to run on.
**compile sdk version** is the the version of Android that the build tools uses to compile & build the application in order to release, run, or debug.
*Usually* the compile sdk version and the target sdk version are the same.

[http://stackoverflow.com/questions/24510219/android-studio-min-sdk-version-target-sdk-version-vs-compile-sdk-version]

## Debugging

Material in textbook chapter 4

Introduce crash (i.e., exception) bug into Asteroids code

### *Strange Build Errors*

Build→Clean/Rebuild Project

Sync Project with Gradle [Tools → Android → Sync Project with Gradle Files]

Check the validity of XML resource files (more applicable to Family Map)

### *Logging*

Show how logcat can be used to view crash stack traces [Android Monitor window, logcat tab]

Exception objects: 1) contain stack traces, and 2) can have a "cause" (original exception). Causes form a linked list of exceptions. The interesting exception is usually the one without a cause (end of the list). The first line in the root exception's stack trace is where the problem originated.

Show how Log class can be used to log messages of different types.

Show how to obtain a stack trace at any point in the code by logging a new Exception().

### *Interactive Debugging*

Debug window

Breakpoints

Click to set.

Run/Resume.  Pause.  Stop.

Step Over.  Step Into.  Step Out.  Run to Cursor.

“View Breakpoints” to view/edit.  “Mute Breakpoints” to disable.

Frames tab: view stack trace when program is stopped

Threads tab: view the program’s threads

Variables tab: view the data in the selected stack frame

Watches: create permanent displays of variable and expression values

Exception breakpoints: Use debugger to catch exceptions where they are thrown

## ADB

Android Debug Bridge (adb) is a versatile command line tool that lets you communicate with an emulator instance or connected Android device.

http://developer.android.com/tools/help/adb.html

Some of the most useful adb commands are:

To list available Android devices: “adb devices”

When a device is not being detected, you can try restarting the adb server by running the following two commands: “adb kill-server” followed by “adb start-server”

To install an app (i.e., a .apk file) on a device: “adb install <path-to-apk-file>”

To copy a file from a device to your development computer: “adb pull <remote-file> <local-file>”

To login directly to the device so you run commands on it: “adb shell”

Add Android SDK’s “platform-tools” subdirectory on your PATH environment variable so you can run adb from a command line.

## Accessing SQLite File on a Device

Adb can be used to access the SQLite file on a device.

Four techniques given in Database lecture slides