CMPS 312 Mobile Application Development

LAB 1: SETTING UP THE ANDROID DEVELOPMENT ENVIRONMENT

Objective

1. Learn the basics of Git and GitHub
2. Set up your Android development environment, familiarize yourself with it, and use it to create and debug simple applications.

Once you've completed this lab, you should have an initial working understanding of the tools developers use when they create Android apps. You should also know how to develop and debug a simple Android application. And how to use git and GitHub version control system to manage and track changes in your code across versions.

Overview

In this lab, you will set up your Android development environment by installing the Android Studio, and several other development tools and documentation. You will use these tools to create and experiment with Android Virtual Devices, the Android emulator, and Android applications.

You will also need to create a a GitHub repository. This repository will hold all your lab work, including in-lab assessments, homework that you will be turned in during the semester.

PART A – GITHUB

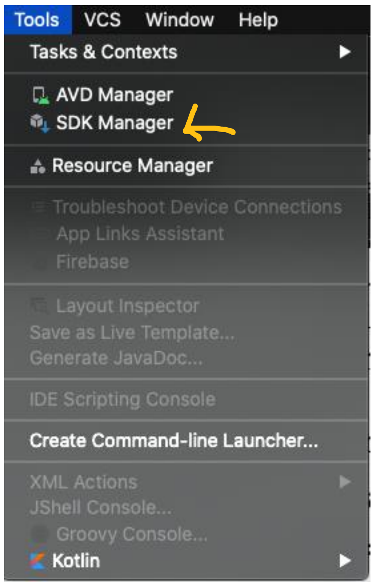
1. Download and Install Github Desktop app from https://desktop.github.com
2. While the download is in progress go to https://github.com and create a Github Account
3. Create a GitHub repository online and clone it
4. Add a new file to your git repo and create your first commit
5. Push your changes to your GitHub repo by using the Github desktop app
6. Create a branch of your master repo and make some modifications to your files
7. Merge the branch to your master repo and push the changes
8. Push a pull request

PART B – DOWNLOADING AND INSTALLING ANDROID STUDIO

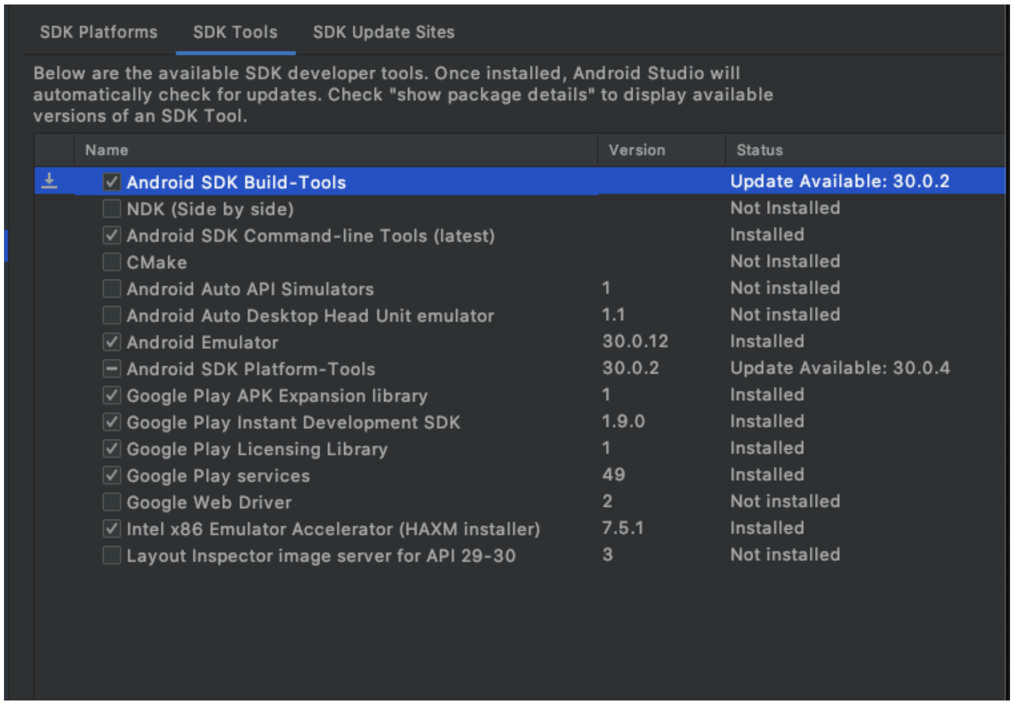
1. Download Android Studio from the Android developer's website
    a. https://developer.android.com/studio/index.html
2. To install Android Studio on **Windows**, launch the **.exe** file you downloaded and follow the setup wizard to install Android Studio and any necessary SDK tools.
3. To install Android Studio on your **Mac**, launch the Android Studio .DMG file. Then Drag and drop Android Studio into the Applications folder, then launch Android Studio. The Android Studio Setup Wizard guides you through the rest of the setup, which includes downloading Android SDK components that are required for the development.

    ➔On the Mac, the path for Android Studio's Java environment can be set with this export command: export JAVA_HOME=/Applications/Android\ Studio.app/Contents/jre/jdk/Contents/Home/

4. Configure SDK Manager OR if you already opened the application, then go to tools➔ Android➔SDK Manager

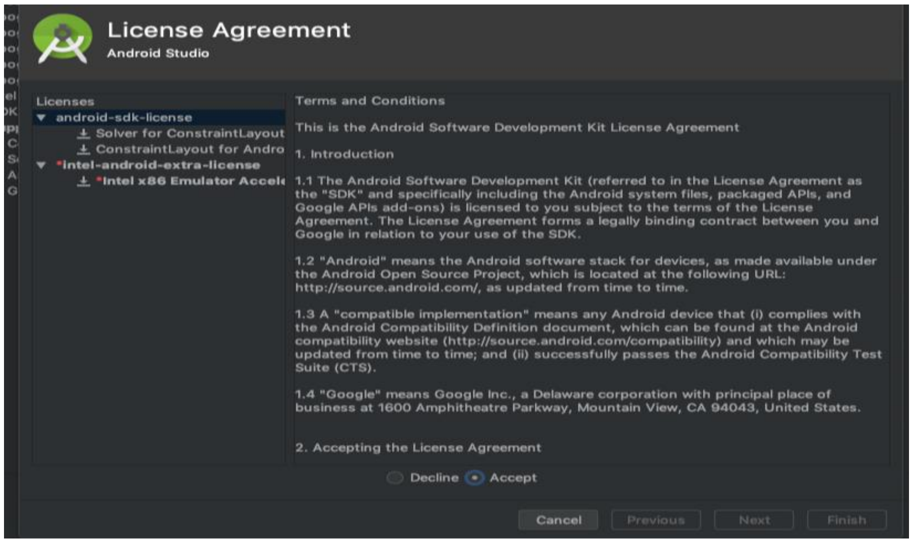


5. Launch SDK Manager and Install the following if they are not installed already.



Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
✓ Android SDK Build-Tools		Update Available: 30.0.2
NDK (Side by side)		Not Installed
✓ Android SDK Command-line Tools (latest)		Installed
CMake		Not Installed
Android Auto API Simulators	1	Not installed
Android Auto Desktop Head Unit emulator	1.1	Not installed
✓ Android Emulator	30.0.12	Installed
− Android SDK Platform-Tools	30.0.2	Update Available: 30.0.4
✓ Google Play APK Expansion library	1	Installed
✓ Google Play Instant Development SDK	1.9.0	Installed
✓ Google Play Licensing Library	1	Installed
✓ Google Play services	49	Installed
Google Web Driver	2	Not installed
✓ Intel x86 Emulator Accelerator (HAXM installer)	7.5.1	Installed
Layout Inspector image server for API 29-30	3	Not installed



License Agreement
Android Studio

Licenses
- android-sdk-license
  - Solver for ConstraintLayout
  - ConstraintLayout for Andro
- *intel-android-extra-license
  - *Intel x86 Emulator Accel

Terms and Conditions

This is the Android Software Development Kit License Agreement

1. Introduction

1.1 The Android Software Development Kit (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.

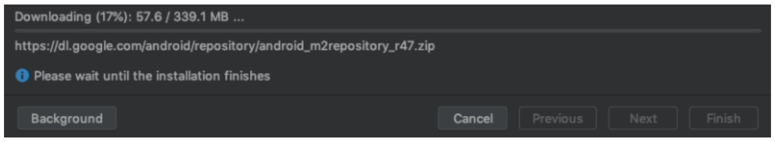
1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: http://source.android.com/, as updated from time to time.

1.3 A "compatible implementation" means any Android device that (i) complies with the Android Compatibility Definition document, which can be found at the Android compatibility website (http://source.android.com/compatibility) and which may be updated from time to time; and (ii) successfully passes the Android Compatibility Test Suite (CTS).

1.4 "Google" means Google Inc., a Delaware corporation with principal place of business at 1600 Amphitheatre Parkway, Mountain View, CA 94043, United States.

2. Accepting the License Agreement

Decline [✓ Accept]



Downloading (17%): 57.6 / 339.1 MB ...

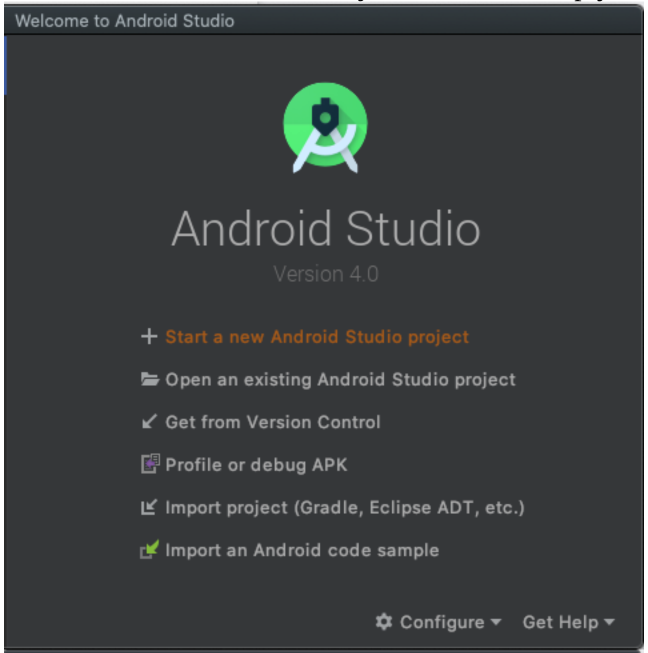
https://dl.google.com/android/repository/android_m2repository_r47.zip

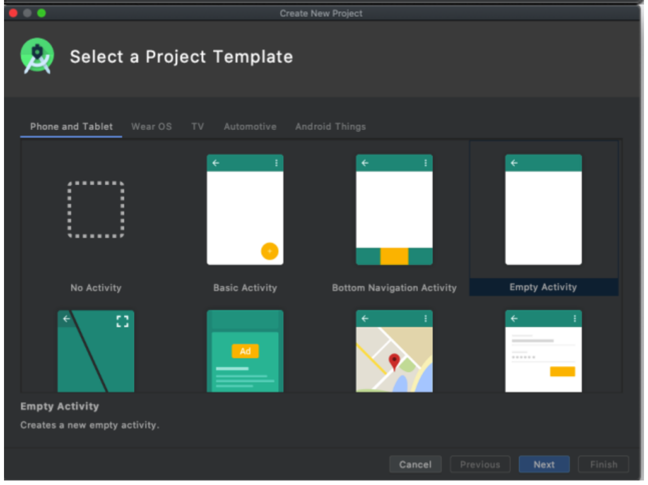
Please wait until the installation finishes

Part C: Creating a simple "hello world" application

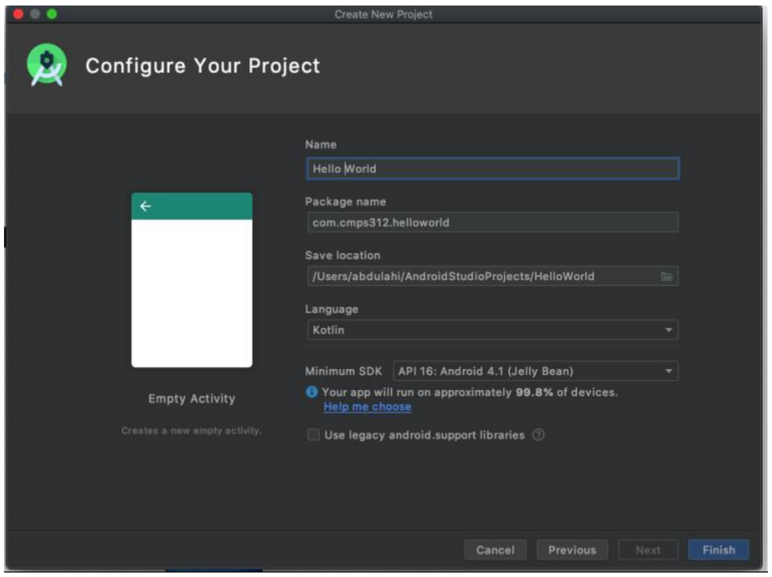
Configuring the project

1. Open Android Studio Application
2. Start New Android Studio Project and select Empty Activity


Welcome to Android Studio
Android Studio
Version 4.0
+ Start a new Android Studio project
Open an existing Android Studio project
Get from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help



3. Make the application name: "Hello World."
4. Package Name: "com.cmps312.helloworld
5. Make sure the language is **Kotlin and not Java**
6. Choose the Minimum SDK 8.0 (Click on Help Me)
7. Finish



Create New Project

Configure Your Project

Empty Activity

Creates a new empty activity.

Name: Hello World

Package name: com.cmps312.helloworld

Save location: /Users/abdulahi/AndroidStudioProjects/HelloWorld

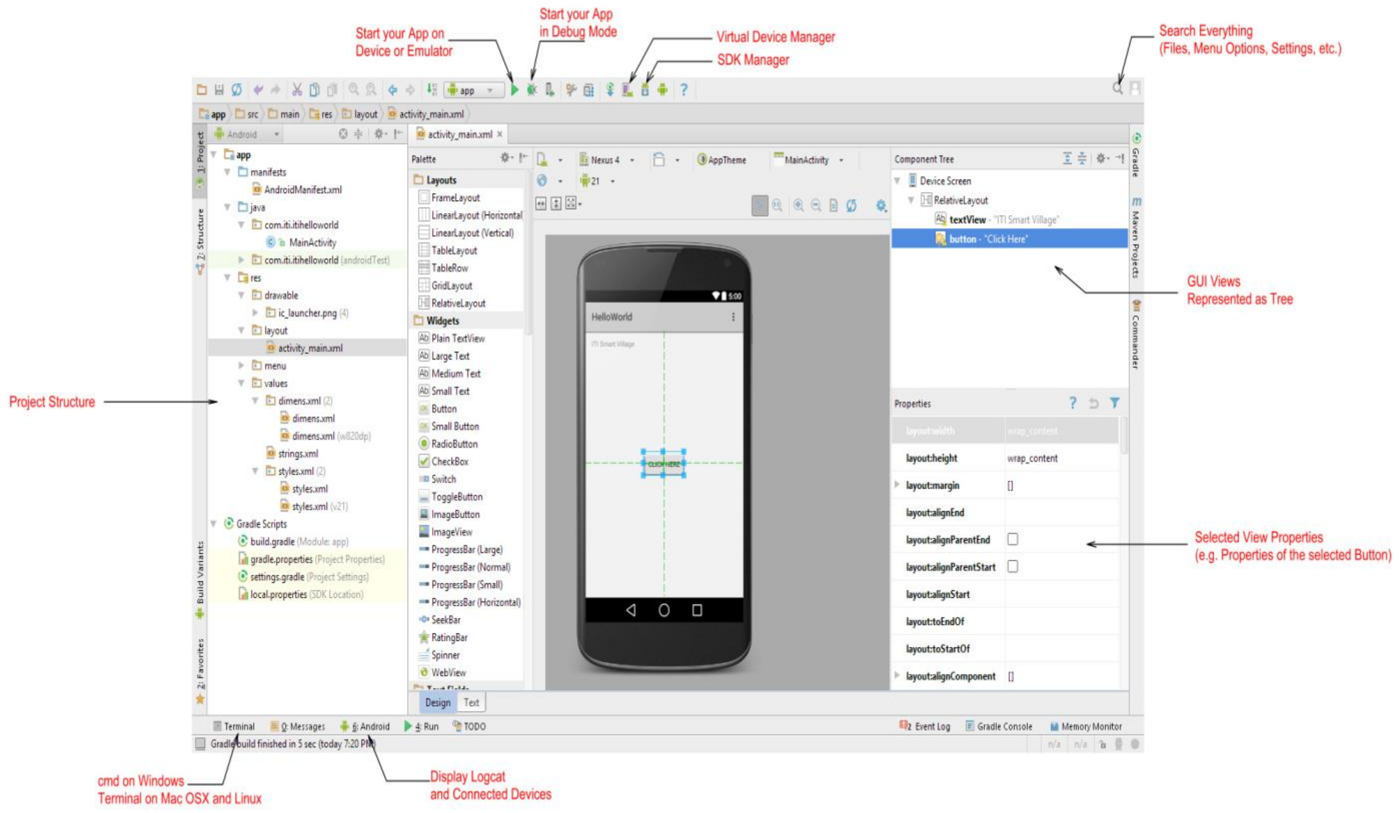
Language: Kotlin

Minimum SDK: API 16: Android 4.1 (Jelly Bean)

Your app will run on approximately **99.8%** of devices.
Help me choose

Use legacy android.support libraries

Cancel | Previous | Next | Finish


Start your App on Device or Emulator
Start your App in Debug Mode
Virtual Device Manager
SDK Manager
Search Everything (Files, Menu Options, Settings, etc.)
GUI Views Represented as Tree
Project Structure
Selected View Properties (e.g. Properties of the selected Button)
cmd on Windows Terminal on Mac OSX and Linux
Display Logcat and Connected Devices

Running the project

A. Using Real Physical Device

1. Plug the device through USB port (Windows automatically downloads your drivers)
2. If a window did not download your device's driver then go to Google "OEM Drivers from android developers" and follow the instructions)
3. Enable the developers option on your physical phone by :
   i. Go to Settings => "About Your device" => "Build Number "then press it Seven Times. This will enable the developer's option
   ii. Search for the developers Options inside the settings
   iii. Tick the USB Debugging Enable box

B. Using Emulator

1. Click on the AVD Manager Button on the top right corner on your android studio
2. At the bottom of the menu, click on "Create Virtual Device"
3. You will get three different choices
   i. You can define new Hardware Profile – Look in the internet
   ii. You can Import Profile from someone else's AVD
   iii. Use the predefined emulators
   iv. Combination (Best)

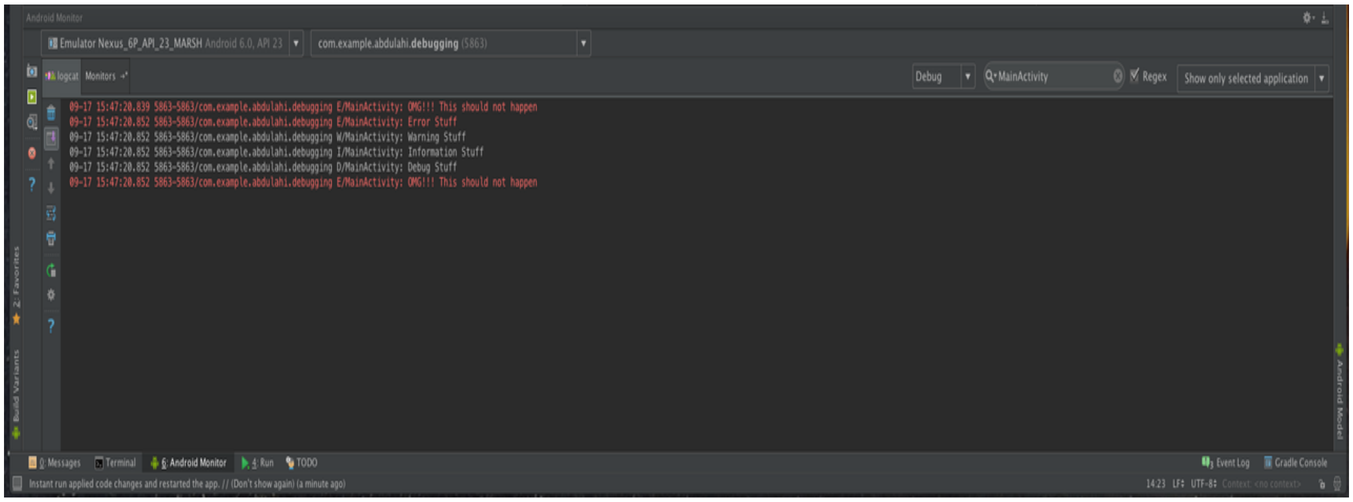
Run the Application by pressing on the RUN button on the top corner

Part C: Debugging Using System Logs

i. Log.v(); // Verbose (If you want to log everything in your app)
ii. Log.d(); // Debug (You want to view some variables values)
iii. Log.i(); // Info (Successfully downloaded the file/ connected)
iv. Log.w(); // Warning (When unexpected thing happened)
v. Log.e(); // Error (Something bad happened ლ(ಠ_ಠლ)
vi. Log.wtf()// (What a Terrible Failure) When something extremely bad happened ヽ(ಠ_ಠ)ノ

Add the above Logs to the Hello World Application

i. val tag = "**MyMainActivity**";
ii. Log."**X**"(**tag**, "**MESSAGE**"); (Replace the X by the above Log type)
iii. Filter the different Log Messages by using the Android Monitor window



```
09-17 15:47:20.839 5863-5863/com.example.abdulahi.debugging E/MainActivity: OMG!!! This should not happen
09-17 15:47:20.852 5863-5863/com.example.abdulahi.debugging E/MainActivity: Error Stuff
09-17 15:47:20.852 5863-5863/com.example.abdulahi.debugging W/MainActivity: Warning Stuff
09-17 15:47:20.852 5863-5863/com.example.abdulahi.debugging I/MainActivity: Information Stuff
09-17 15:47:20.852 5863-5863/com.example.abdulahi.debugging D/MainActivity: Debug Stuff
09-17 15:47:20.852 5863-5863/com.example.abdulahi.debugging E/MainActivity: OMG!!! This should not happen
```

Instant run applied code changes and restarted the app. // (Don't show again) (a minute ago)

Capturing Screenshots

i. Run your App in Debug Mode.
ii. Click Logcat at the bottom of the IDE.
iii. On the left corner expand the two arrows and Click Screen Capture on the left.
iv. Optional: To add a device frame around your screenshot, click Frame screenshot.
v. Click Save.

Capturing Video

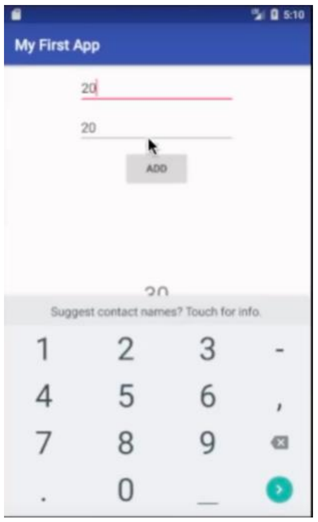
i. Run your App in Debug Mode.
ii. Click Logcat at the bottom of the IDE.
iii. Click Screen Record on the left.
iv. Click Start Recording.
v. Interact with your app.
vi. Click Stop Recording.
vii. Enter a file name for the recording and click **OK**.

PART D: Handling events [Buttons]

There are 2 ways to handle the click event in a button

- Onclick in XML layout
- Using an OnClickListener

Let us create the following simple app that takes two numbers as an input and displays their result on the screen.



1. Create an app and name it "Simple Calculator."
2. Modify the layout to look like the image above. It should have
   a. Two EditText boxes with type number,
   b. An Add Button and
   c. A TextBox that displays the result once the user clicks on the Add
3. Run your application on the emulator.