1. Getting Started (Initial Setup)
   1. Android Studio Installation
      1. Install Android Studio with android development tools. You can find the installer at <http://developer.android.com/sdk/installing/index.html?pkg=studio>
      2. Extract studio by installing the downloaded installer.
      3. Go to the studio folder and start Android Studio.
      4. On some Windows systems, the launcher script does not find where Java is installed. If you encounter this problem, you need to set an environment variable indicating the correct location.

Select **Start menu > Computer > System Properties > Advanced System Properties**. Then open **Advanced tab > Environment Variables** and add a new system variable JAVA\_HOME that points to your JDK folder, for example: C:\Program Files\Java\jdk1.7.0\_21

* 1. SDK Manager

On the top panel you will see two android specific options. Hover over them and click on the SDK manager. Here are different components that you should install from the SDK manager.

* + 1. SDK Tools: These are some tools required for doing Android development. This includes tools like SDK manager and Virtual device manager.

* + 1. SDK Platform Tools: Important tools like adb that are required for debugging a lot of issues related to android devices and software.
    2. SDK Build Tools: These tools are required to build, run and test android applications.
    3. Android API Specific Packages

In the SDK manager you will see different installable for different Android API levels. Here are the different packages that you find inside each API Level.

* + - 1. SDK Platform: This is the main package that you need to test your app against a particular api level.
      2. System Images: You will find multiple system images for each api. These images are used to emulate a specific API level on the emulator machine. You would need to install this along with the platform if you want to test your code using an emulator for a particular android api level.
      3. Sources for SDK: This has the source code for the particular SDK.
      4. Google API’s: You will have to install these in case you want your emulator to have google apps like Google maps etc.
      5. Samples: Includes sample code files showing example use of different api’s from that API level.
    1. Extras
       1. Android Support Library: This includes some additional API’s and it also provides backward compatibility by providing support for newer api’s in older Android releases.
       2. Google Play Services: Allows your app to access Google maps API, Google Wallet etc.
       3. HAXM (Hardware Accelerated Execution Manager): is an engine created by Intel to accelerates the Android emulation. This tool leads to a considerable improvement in the speed of emulator and is the prime factor why we should prefer intel based system images.
  1. Emulator & Virtual Device Manager

You need to setup an emulator to test your application without using a physical device. Even if you have a physical device to test your app on, you will have to use emulators to test on all different API levels and devices, which your app supports. Virtual Device Manager (VDM)’s icon is right next to the SDK Manager icon.

Here are the different fields that you will have to set while creating a emulator from the Virtual Device Manager:

* + 1. Device: Lets you choose the device you want to emulate. This lets you emulate your app for a screen as small as 2.7” to a 10” inch tablet. You can even test your app on Android TV.
    2. Target SDK: Here you decide the android version (API Level) that should be installed on the emulated device. You can only install an API level if you have downloaded the corresponding SDK in the SDK manager.
    3. CPU (System Image): You can decide whether your emulator should behave like it’s running on an ARM processor or an Intel processor. You might see more or less processor options depending on the API level. You should prefer the Intel images because of the benefit that HAXM provides. In the dropdown you will only see those options, which you have downloaded using the SDK manager.
    4. Emulator Options:
       1. Snapshot: Emulators take a lot of time booting up everytime you open them. Snapshot option lets you avoid this long wait after the first load. While creating an emulator if you tick this option, a snapshot of emulator will be taken everytime you close the emulator and next time when you start the emulator this snapshot will be used to quickly boot it up.
       2. Use Host GPU: This is an important checkbox which use should almost always tick. This option accelerates the emulator by letting it use the GPU in your computer. This option can’t be used when you are using the Snapshot option.
    5. Other Options: There are few other options, which let you decide various specs for the emulated device. E.g.
       1. Keyboard: Choose this option if you want your emulated device to have a hard keyboard.
       2. Skin: Different looks that can be given to the emulator.
       3. Camera: You can either emulate a camera or even use your computer’s camera to act as front or back camera of the emulator. This will allow you to test apps, which require camera usage.
       4. Memory: Maximum memory that can be allocated to the emulator. Make sure this value is not more than the value you mentioned for the HAX, in which case you will see an error saying that HAX couldn’t be initiated.
       5. Internal Storage: Amount of storage you would require to test your app. This will be the amount of internal storage available to you in the emulated device.
       6. SD Card: You can emulate external storage using this option.
  1. Installing on Android Hardware: To install the app on your Android device you would have to enable it for development.

Here are the steps to do so:

* + 1. Enabling developer options: You will find your developer options in Settings. For Android 4.2 and higher, developer settings are not visible. To access these settings you will have to go to Settings🡪About Phone and tap Build number seven times.
    2. Enabling USB Debugging: Enable USB debugging inside the developer options to use the device for testing.
    3. Drivers: In case you can’t connect your device with Eclipse, you might have to install drivers to make your device work, search online and install the required drivers.

1. Creating your First App: You can create an Android application by selecting File 🡪 New 🡪 Android Application Project. Here are few settings on the first page, which you should understand. After this screen keep pressing next. You will be able to understand these settings later on.
   1. Different Settings
      1. Minimum Required SDK: This setting lets you decide what is the minimum API level that your app is going to support. You will be able to target more users if you keep this number lower but that would also mean that you would have to write cumbersome code to make your app compatible with old API levels. You should keep the following stats in mind before making this decision: Almost no one is using anything below Android2.2, 20% users are on Gingerbread (2.3). Remaining 80% of users are using > Android4.0.
      2. Target: This is the maximum API level that you are planning to support with your app. This should almost always be the latest level. This setting makes sure that whenever a new OS version is provided to the users, developers would test their apps with the new OS and make sure that no fixes are required to make the app work with the new operating system.
      3. Compile with: This is the SDK, which is used to build your app. This should be the latest SDK. Latest SDK comes preinstalled with Eclipse (ADT), in case you want to compile with some other SDK you must have installed that SDK level from the SDK manager.
   2. Different files and folders: After creating the project you would see that a bunch of files and folders are auto generated for you. Here are the most important files you need to look at.
      1. Manifest.xml: This is one of the most important files in the project. Your application will communicate to the operating system through this file. This file has the settings like minimum SDK version etc, which you chose while creating the project. This is where you mention the launcher icon, app name and a bunch of other settings.
      2. Source Code (src): This is the folder where all your java code goes. By default you will a MainActivity.java class in this folder. We will look at what an activity means later in this course.
      3. Resources (res): All the supporting files go in the res folder. This has a bunch of sub folders for different type of resources.
         1. Drawable: This folder contains various images that we will use in our application. By default we have an image (ic\_launcher), which will be our launch icon. You will see that we have a bunch of drawable folders like –hdpi, -ldpi, -mdpi etc. These folders let you provide different images for different variants of Android screens. High def screens will require larger images compared to low def screens as high def screens have more pixels per inch of screen.
         2. Layout: This folder contains the UI code. Files inside this folder define how your application is going to look like.
         3. Menu: This folder contains xml files, which define what items should go into the menu corresponding to various screens.
         4. Values: This folder contains files for different kind of components that you shouldn’t hard code in your Java code. E.g. instead of hard coding “Coding Blocks” string inside an app, we can add this string in Strings.xml file and later on access it in the Java files. This way you don’t have to go through the Java files to change any of these values later.
   3. Installing the app: Just use the Run button from the panel on the top or you can right click on your project and choose Run as🡪Android application. You will get options to run the app on the physical device (if any attached) or one of the emulators.
2. Android Studio Shortcuts
   1. Double Shift: Search anywhere
   2. Ctrl + click: Go to declaration of the object
   3. Find Usage
   4. Ctrl + Alt + L: Format Code
   5. Generate constructors and getters and setters
3. Gradle File Structure:
   1. Module version controls the overall properties and is not to be changed
   2. App version contains properties specific to our application and this is where third party libraries are added, support and other properties are defined
4. Homework
   * 1. Install Android Studio and install required packages from SDK manager.
     2. Create a new Android app project and make sure you are able to install it on the emulator as well as on a physical device.