Catanduanes State University
College of Information and Communications Technology
Virac, Catanduanes

LEARNING MATERIALS AND COMPILATION OF LECTURES/ACTIVITIES

ITRACKA2

Platform Technologies: Mobile Application Development

DISCLAIMER

This learning material is used in compliance with the flexible teaching-learning approach espoused by CHED in response to the pandemic that has globally affected educational institutions. Authors and publishers of the contents are well acknowledged. As such, the college and its faculty do not claim ownership of all sourced information. This learning material will solely be used for instructional purposes not for commercialization.

CatSU College of Information and Communications Technology

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION TO MOBILE APPLICATION

LEARNING OUTCOMES

At the end of this chapter, the students shall be able to:

1. Discuss the characteristics of mobile application and its usage.
2. Evaluate needed components for android application development.
3. Set up android environment for development.

KEY TERMS

1. Android
2. Android Software Development Kit
3. Java Development Kit (JDK)
4. Java
5. Android Studio
6. Android Virtual Device (AVD)

LESSON 1: CHARACTERISTICS OF MOBILE APPLICATION

1.1. WHAT IS A MOBILE APPLICATION?

- ✓ A mobile app is a software program you can download and access directly using your phone or another mobile device through their specific app stores such as google play store for android, etc.
- ✓ Software stack for mobile devices such as mobile phones, tablets, TVs, and wearables.

As per the website, https://www.consumer.ftc.gov/articles/0018-understanding-mobile-apps:

In order to download and use an app, you need a smart phone or another mobile device with internet access. Not all apps work on all mobile devices. Once you buy a device, you're committed to using the operating system and the type of apps that go with it. The Android, Apple, Microsoft, Amazon, and BlackBerry mobile operating systems have app stores online where you can look for, download, and install apps. Some online retailers also offer app stores. You'll have to use an app store that works with your device's operating system. To set up an account, you may have to provide a credit card number, especially if you're going to download an app that isn't free.

Some apps are distributed for free through app stores, the developers make money in a few ways:

- ✓ Some sell advertising space within the app. The app developers can earn money from the ads, so they distribute the app for free to reach as many users as possible.
- ✓ Some apps offer their basic versions for free. Their developers hope you'll like the app enough to upgrade to a paid version with more features.

- ✓ Some apps allow you to buy more features within the app itself. Usually, you are billed for these in-app purchases through the app store. Many devices have settings that allow you to block in-app purchases.
- ✓ Some apps are offered free to interest you in a company's other products. These apps are a form of advertising.

1.2. TYPES OF DATA AN APP CAN ACCESS

When you sign up with an app store or download individual apps, you may be asked for permission to let them access information on your device. Some apps may be able to access:

- ✓ your phone and email contacts
- ✓ call logs
- ✓ internet data
- ✓ calendar data
- ✓ data about the device's location
- ✓ the device's unique IDs
- ✓ information about how you use the app itself

Some apps access only the data they need to function; others access data that's not related to the purpose of the app.

Remember that someone may be collecting data on the websites you visit, the apps you use, and the information you provide when you're using the device – whether it's the app developer, the app store, an advertiser, or an ad network. And if they're collecting your data, they may share it with other companies. If you are concerned about how your information is being shared, check the "privacy" settings on your device or look for ways to "opt-out" of data collection in the app privacy policy.

1.3. THE THREE GIANT VENDORS OF MOBILE APPLICATIONS (PANHALE, 2015)

1. **APPLE**

The Apple platform is a big ecosystem. As you may know, the iPhone, iPad, and MacBook all fall under the Apple ecosystem. These devices benefit from being in the same ecosystem, as a single store distributes iOS-based applications. Apple verifies applications and grants permission for sale based on its guidelines for acceptance.

Apple also promotes development of applications targeting the Apple ecosystem, by offering many common APIs and a common approval process. Development is made easy by Objective C and the Xcode IDE, along with many APIs to natively access (meaning at the device level) features in the app such as a camera and location.

2. **GOOGLE (ANDROID)**

The key thing to understand is that the Google ecosystem is different. You will have to consider Android-based devices. The Android OS is an open platform to manufacturers, so the market has many device manufactures compared to Apple. Because Android is allowed to customize, this only boosts the variety of available Android devices.

Because of the multiple device manufacturers, ultimately many devices vary in size, resolution, and available features, so this ecosystem has many challenges compared to Apple. For example, if you develop an application for a Samsung Galaxy Android-based phone, changes may be required if the application is ported to a Google Nexus device, because of resolution differences.

The good point in favor of Android is that unlike Apple, application development for Android- based devices is mostly in Java. This is one of the popular and older languages, so it is easy to find programmers in Java for Android, compared to Objective C or Xcode for Apple.

3. **MICROSOFT**

The Microsoft ecosystem is similar to that of Apple. While working in the Microsoft ecosystem, you have to consider a range of devices, including the Windows desktop, Windows Phone, and Surface. Development platforms are the .NET Framework and XNA. Microsoft has the Windows App Store for distributing applications. The preferred IDE is Microsoft Visual Studio. Mainly C# and VB.NET are the languages used for development. The process and ratio of sharing benefits remains the same as that of Apple (70:30, in favor of the application owner).

As we develop mobile applications, you have to consider a lot of factors in order for our projects to be successful. You cannot simply just develop applications without considering your target audience, or whether it has objectives or not. You need to understand the characteristics of a mobile application, and use it as you develop your own mobile app in the future.

As we develop mobile applications, you have to consider a lot of factors in order for our projects to be successful. You cannot simply just develop applications without considering your target audience, or whether it has objectives or not. You need to understand the characteristics of a mobile application, and use it as you develop your own mobile app in the future.

1.4. CHARACTERISTICS OF MOBILE DEVICES (SALMRE, 2005)

1. **USAGE PATTERNS**

Mobile device applications tend also to be more focused on enabling a few specific features very well as opposed to offering the general-purpose exploratory environment that successful desktop applications do. Because mobile devices are often operated using a single hand or by tapping a small screen with a stylus, it is important that users of the device be able to quickly discover and navigate to the information and features they want. The ability to quickly navigate to a small set of key features is an important aspect of a great mobile device experience.

2. **FORM FACTOR**

The need to fit comfortably into one's pocket is a key defining characteristic of most mobile devices. This physical constraint is the basis for the mobile device's utility; if it fits in your pocket, it is mobile. Some additional important form-factor considerations are as follows:

- ✓ The ability to be used in crowded and noisy spaces
- ✓ Single- or two-handed operation
- ✓ No power cords or communications cables for long periods of time

3. RELIABILITY REQUIREMENTS

With regard to reliability requirements, mobile devices paradoxically resemble servers more than they do desktops. The reasons for this are as follows:

- ✓ Much like servers, mobile devices and their applications are often left running 24 hours a day, 7 days a week.
- ✓ Much like servers, mobile device applications have to deal effectively with unexpected failures.
- ✓ Much like servers, mobile device operating systems and applications often do not use memory paging files.
- ✓ Many mobile devices serve other critical purposes while running foreground applications.

1.5. CHARACTERISTICS OF A SUCCESSFUL MOBILE APPLICATION

1. THE IDEA

Behind every great app is a great idea. But make no mistake, a great idea does NOT guarantee a great app. The most successful apps are those that identify and solve a problem, whether that be communicating with international family or friends, like WhatsApp, or conveniently checking in for a flight, like American Airlines.

2. IDENTIFIES AS TARGET DEMOGRAPHIC

When you take the time to understand who will be using your app, you have a better chance of creating an end-product that people will actually use. The benefits of conducting user research are endless. If you conduct user research you can create a solution that is tailored to the specific needs of your target audience. Learn about your users' pain points, age, gender, interests, behaviors and solutions they currently use to solve the problem you are trying to remediate. Gathering this intelligence will allow you to create a successful app that aligns with their needs and expectations.

3. ENCOURAGES USER ENGAGEMENT

A successful app is one that continuously engages users. An intuitive and seamless onboarding process will create a positive first impression that encourages future engagement. A complex onboarding process increases mobile app abandonment rate. You can also implement an incentive-based system that rewards users for visiting your app.

When used correctly, push notifications can be an extremely helpful tool in driving user engagement. Push notifications can help to re-engage inactive users and encourage users to finish incomplete tasks. Including elements of personalization improves the overall user experience and leads to increased engagement as a result.

4. FOLLOWS PLATFORM DESIGN GUIDELINES

Your mobile app should be platform-appropriate. Google Play and Apple App Store both lay out explicit design guidelines for app designers and developers to follow. Catering the design of your mobile app to the specific device type will ensure that the user experience is seamless. Once you determine which device type is most popular among your target audience, you can focus on first designing for that specific platform.

5. USER INTERFACE DESIGN

User interface design attracts users to your mobile app and provides an opportunity to present your captivating content. Your user interface design should be simple to navigate, but unique features will set you apart from competition. It is important that you remain consistent with design throughout the entire app. Keeping elements like font size, font family and color scheme uniform across the app will improve the visual appearance and assist with the comprehension of content.

6. USE OF FAMILIAR SCREENS

Similarly, while your app should have unique aspects, it should not be so unique that no one can operate it. Using familiar screens and gestures, like "What's New" or "Profile" will

improve the usability of your app. You can avoid the general learning curve associated with a new app by employing screens users are already comfortable with.

7. **FRICTIONLESS NAVIGATION FEATURE**

   Your mobile app should include a simple and intuitive navigation and/or search feature – especially if you have a high volume of content. Users generally will not spend extra time searching for content that is not easily accessible to them. Incorporating features that simplify their experience will increase user engagement. Navigating your app should be natural and obvious to the user.

8. **RESPONSIVE**

   Speed should be a top priority while creating a mobile app. A successful app must always respond instantly to user input, or at minimum let the user know that the app itself is waiting. An app with a slow response time will degrade the user experience and will likely be deleted. Users expect lightning fast results wherever and whenever they need them.

SUPPLEMENTARY LEARNING RESOURCES

Characteristics of a successful mobile app. (2019, June 12). Seamgen Blog. https://www.seamgen.com/blog/5-characteristics-of-a-successful-app/

Salmre, I. (2005). *Writing mobile code: Essential software engineering for building mobile applications*. Addison-Wesley Professional.

Understanding mobile apps. (2018, March 13). Consumer Information. https://www.consumer.ftc.gov/articles/0018-understanding-mobile-apps#basics

SELF-ASSESSMENT QUESTION 1-1

What are the eight (8) characteristics of a successful mobile app? Explain briefly.

LESSON 2: THE ANDROID VIRTUAL MACHINE

Android Virtual Machine has the similar functionality as the Java Virtual Machine, but not exactly the same. Each android application runs in its own Android Virtual Machine, which means that each app runs in isolation. It also runs as a separate Linux process. Android executable have an extension of dex, which is optimized for minimal memory footprint. It was used to be called Dalvik VM, but was replaced by the Android Runtime (ART) in Lollipop.

2.1. ANDROID OS

Android is an open source**,** Linux-based software stack created for a wide array of devices and form factors. For example, the **Android Runtime (ART)** relies on the Linux kernel for underlying functionalities: networking, multi-threading and low-level memory management.

The Android operating system is a mobile operating system that was developed by Google to be primarily used for touchscreen devices, cell phones, and tablets. Its design lets users manipulate the mobile devices intuitively, with finger movements that mirror common motions, such as pinching, swiping, and tapping. Google also employs Android software in televisions, cars, and wristwatches—each of which is fitted with a unique user interface.

The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008. The source code for Android is available under free and open source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.

2.1.1. WHY ANDROID USES VIRTUAL MACHINE?

There are many reasons that Google engineers decide to use Android with VM, but two main reason is:

- **Security**: In theory, app code is totally isolated by the VM and cannot even “see” the host OS, so app code that contains malware cannot affect system directly, make app and system more robust and reliable.
- **Platform independent**: Android platform can run on different devices with different architectures (ARM, MIPs, x86). To abstract out the need to compile binaries for each architecture, VM comes into play.

2.2. ANDROID RUNTIME (ART) AND DALVIK

Android runtime (ART) is the managed runtime used by applications and some system services on Android. ART and its predecessor Dalvik were originally created specifically for the Android project. ART as the runtime executes the Dalvik Executable format and Dex bytecode specification.

ART and Dalvik are compatible runtimes running Dex bytecode, so apps developed for Dalvik should work when running with ART. However, some techniques that work on Dalvik do not work on ART.

2.2.1. ART FEATURES

Here are some of the major features implemented by ART.

1. **AHEAD-OF-TIME (AOT) COMPILATION**

   ART introduces ahead-of-time (AOT) compilation, which can improve app performance. ART also has tighter install-time verification than Dalvik.

   At install time, ART compiles apps using the on-device **dex2oat** tool. This utility accepts DEX files as input and generates a compiled app executable for the target device. The utility should be able to compile all valid DEX files without difficulty. However, some post-

processing tools produce invalid files that may be tolerated by Dalvik but cannot be compiled by ART.

2. IMPROVED GARBAGE COLLECTION

Garbage collection (GC) can impair an app's performance, resulting in choppy display, poor UI responsiveness, and other problems. ART improves garbage collection in several ways:

- One GC pause instead of two
- Parallelized processing during the remaining GC pause
- Collector with lower total GC time for the special case of cleaning up recently-allocated, short-lived objects
- Improved garbage collection ergonomics, making concurrent garbage collections more timely, which makes GC_FOR_ALLOC events extremely rare in typical use cases
- Compacting GC to reduce background memory usage and fragmentation

3. DEVELOPMENT AND DEBUGGING IMPROVEMENTS

ART offers a number of features to improve app development and debugging.

Support for sampling profiler

Historically, developers have used the Traceview tool (designed for tracing application execution) as a profiler. While Traceview gives useful information, its results on Dalvik have been skewed by the per-method-call overhead, and use of the tool noticeably affects run time performance.

ART adds support for a dedicated sampling profiler that does not have these limitations. This gives a more accurate view of app execution without significant slowdown. Sampling support was added to Traceview for Dalvik in the KitKat release.

Support for more debugging features

ART supports a number of new debugging options, particularly in monitor- and garbage collection-related functionality. For example, you can:

- See what locks are held in stack traces, then jump to the thread that holds a lock.
- Ask how many live instances there are of a given class, ask to see the instances, and see what references are keeping an object live.
- Filter events (like breakpoint) for a specific instance.
- See the value returned by a method when it exits (using “method-exit” events).
- Set field watchpoint to suspend the execution of a program when a specific field is accessed and/or modified.

4. IMPROVED DIAGNOSTIC DETAIL IN EXCEPTIONS AND CRASH REPORTS

ART gives you as much context and detail as possible when runtime exceptions occur. ART provides expanded exception detail for:

- `java.lang.ClassCastException`
- `java.lang.ClassNotFoundException`
- `java.lang.NullPointerException`.
- `java.lang.ArrayIndexOutOfBoundsException`
- `java.lang.ArrayStoreException`

For example, `java.lang.NullPointerException` now shows information about what the app was trying to do with the null pointer, such as the field the app was trying to write to, or the method it was trying to call. Here are some typical examples:

```
java.lang.NullPointerException: Attempt to write to field 'int
android.accessibilityservice.AccessibilityServiceInfo.flags' on a null object
reference
```

```
java.lang.NullPointerException: Attempt to invoke virtual method
'java.lang.String java.lang.Object.toString()' on a null object reference
```

ART also provides improved context information in app native crash reports, by including both Java and native stack information.

SUPPLEMENTARY LEARNING RESOURCES

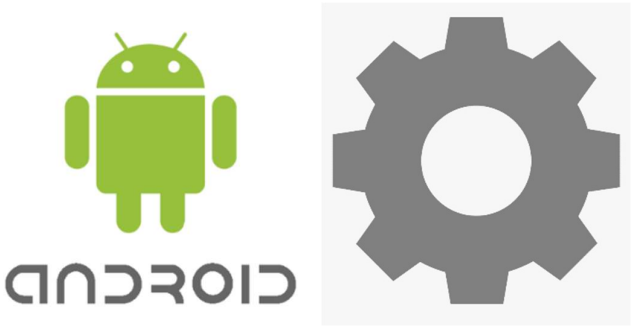
Android - Application components. (n.d.). RxJS, ggplot2, Python Data Persistence, Caffe2, PyBrain, Python Data Access, H2O, Colab, Theano, Flutter, KNime, Mean.js, Weka, Solidity. https://www.tutorialspoint.com/android/android_application_components.htm

Android runtime (ART) and Dalvik. (n.d.). Android Open Source Project. https://source.android.com/devices/tech/dalvik

Android operating system: What you need to know. (n.d.). Investopedia. https://www.investopedia.com/terms/a/android-operating-system.asp

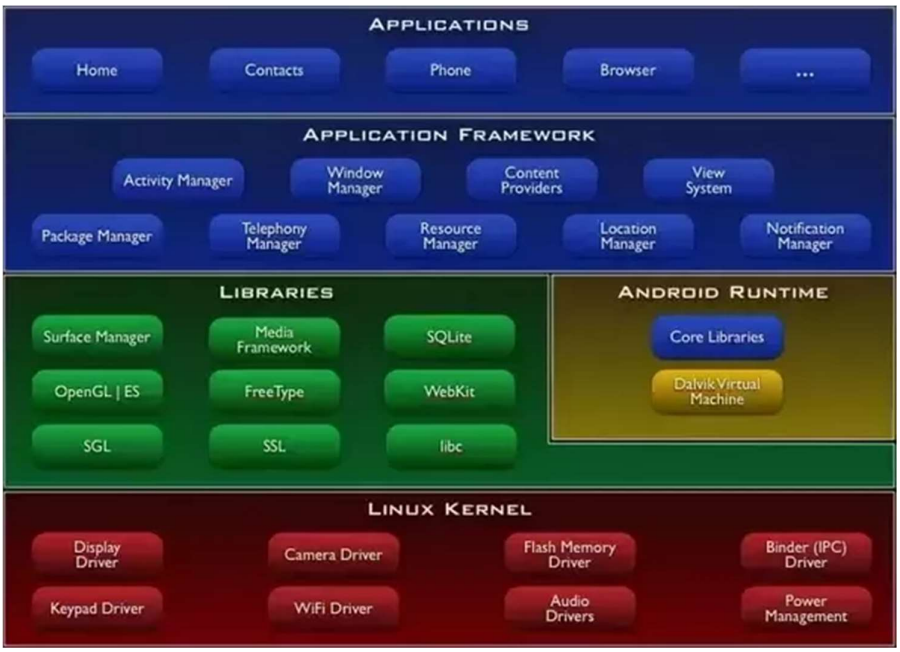
Doan, T. H. (2020, February 3). *Virtual machine in Android: Everything you need to know*. Medium. https://android.jlelse.eu/virtual-machine-in-android-everything-you-need-to-know-9ec695f7313b

LESSON 3: ANDROID APPLICATION COMPONENTS



Android applications are usually developed in the Java language using the Android Software Development Kit or SDK. Once developed, Android applications can be packaged easily and sold out either through a store such as Google Play, SlideME, Opera Mobile Store, Mobango, F-droid and the Amazon Appstore. Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast. I'm pretty sure most of you are using Android as your device's OS.

3.1. ANDROID ARCHITECTURE



1. LINUX KERNEL

At the bottom of the layers is Linux - Linux 3.6 with approximately 115 patches. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

2. LIBRARIES

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

3. ANDROID LIBRARIES

This category encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the application framework libraries in addition to those that facilitate user interface building, graphics drawing and database access. A summary of some key core Android libraries available to the Android developer is as follows –

- **android.app** − Provides access to the application model and is the cornerstone of all Android applications.
- **android.content** − Facilitates content access, publishing and messaging between applications and application components.
- **android.database** − Used to access data published by content providers and includes SQLite database management classes.
- **android.opengl** − A Java interface to the OpenGL ES 3D graphics rendering API.
- **android.os** − Provides applications with access to standard operating system services including messages, system services and inter-process communication.
- **android.text** − Used to render and manipulate text on a device display.
- **android.view** − The fundamental building blocks of application user interfaces.
- **android.widget** − A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
- **android.webkit** − A set of classes intended to allow web-browsing capabilities to be built into applications.

Having covered the Java-based core libraries in the Android runtime, it is now time to turn our attention to the C/C++ based libraries contained in this layer of the Android software stack.

4. ANDROID RUNTIME

As the in the previous lesson, Android runtime (ART) is the managed runtime used by applications and some system services on Android. ART and its predecessor Dalvik were originally created specifically for the Android project. ART as the runtime executes the Dalvik Executable format and Dex bytecode specification.

5. APPLICATION FRAMEWORK

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

The Android framework includes the following key services –

- ✓ **Activity Manager** − controls all aspects of the application lifecycle and activity stack.
- ✓ **Content Providers** − allows applications to publish and share data with other applications.
- ✓ **Resource Manager** − provides access to non-code embedded resources such as strings, color settings and user interface layouts.
- ✓ **Notifications Manager** − allows applications to display alerts and notifications to the user.
- ✓ **View System** − an extensible set of views used to create application user interfaces.

6. APPLICATIONS

You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, Games, etc.

3.2. ANDROID APPLICATION COMPONENTS

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file `AndroidManifest.xml` that describes each component of the application and how they interact.

1. **ACTIVITIES**

   Represents a single screen with a user interface. You write one by subclassing the `Activity` class. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

2. **INTENTS**

   These are events, or messages sent to other application components. It activates Activities, Services, and Broadcast Receivers. You can actually broadcast an intent, and have it handled by another application. For example, your app can broadcast an intent to open a web page, and Android will run the browser. Android prompts user which one he/she wants to use. These are created with an `Intent` object.

3. **SERVICES**

   Services are processes that runs in the background that do not have any UI components. Examples are music playing in the background, downloading data, etc. You can write one by extending the `Service` class.

4. **BROADCAST RECEIVERS**

   Similar to event-handlers, broadcast receivers respond to a system-wide events called “broadcasts” or “intents”. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action. It does not have a UI but can create status bar notifications by using `NotificationManager`. You write one by extending `BroadcastReceiver` class.

5. **CONTENT PROVIDERS**

   These are interfaces for sharing data between applications on request. Such requests are handled by the methods of the `ContentResolver` class. The data may be stored in the file system, the database or somewhere else entirely. A content provider is implemented as a subclass of `ContentProvider` class and must implement a standard set of APIs that enable other applications to perform transactions.

SUPPLEMENTARY LEARNING RESOURCES

Android - Architecture. (n.d.). RxJS, ggplot2, Python Data Persistence, Caffe2, PyBrain, Python Data Access, H2O, Colab, Theano, Flutter, KNime, Mean.js, Weka, Solidity. https://www.tutorialspoint.com/android/android_architecture.htm

Android - Application components. (n.d.). RxJS, ggplot2, Python Data Persistence, Caffe2, PyBrain, Python Data Access, H2O, Colab, Theano, Flutter, KNime, Mean.js, Weka, Solidity. https://www.tutorialspoint.com/android/android_application_components.htm

Developing Mobile Applications with Android. (2015). Active Learning, Inc.

LESSON 4: ANDROID ENVIRONMENT SETUP

Nowadays, it's not that hard and expensive to develop mobile application. All the required tools to develop Android apps are freely available and can be downloaded from the internet. Here is the list of software you will need before you start your Android application programming.

- ✓ Java JDK5 or later version
- ✓ Android IDE: Android Studio

4.1. ANDROID STUDIO

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Apply changes to push code and resource changes to your running app without restarting your app
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

4.1.2. MINIMUM SYSTEM REQUIREMENTS FOR ANDROID STUDIO

For OS, you'll need:

- Microsoft Windows 7/8/10 (64-bit); or
- macOS 10.10 (Yosemite or higher); or
- Linux (Gnome or KDE Desktop), Ubuntu 14.04 or higher (64-bit but capable of running 32-bit applications)
- GNU C Library (glibc 2.19 or later) if you're on Linux

For hardware, you'll need to have at least:

- 4GB RAM (8GB or more recommended)
- 2GB of available HDD space
- 1280 x 800 minimum screen resolution

4.2. INSTALLING THE SOFTWARE DEVELOPMENT KIT (SDK)

1. Download and install the Java Development Kit (JDK) from this link:

https://www.oracle.com/java/technologies/javase-downloads.html

Choose the right installer for your personal computer.

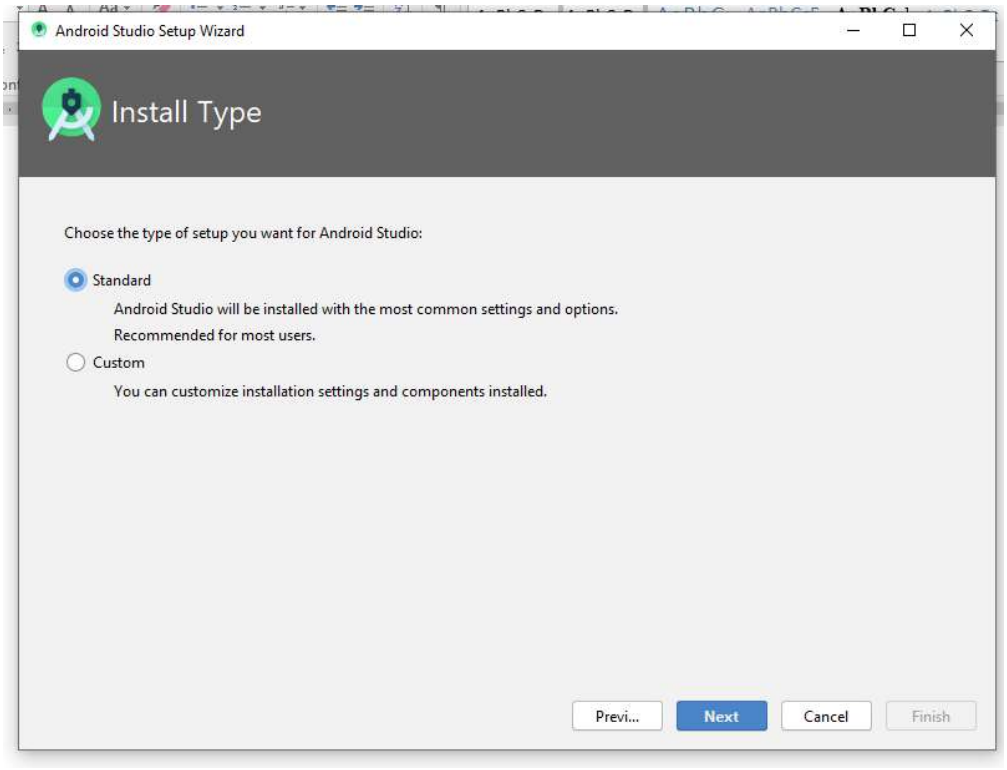
2. If you're running a 64-bit Windows, make sure to download the 64-bit version of the JDK. You will find instructions for installing JDK in downloaded files, you just have to follow the given instructions to install and configure the setup.

3. Set the JAVA_HOME environment variable to point to where the JDK is installed. If you are running Windows and installed the JDK in C:\jdk1.8.0_102, you would have to put the following line in your C:\autoexec.bat file.

```
set PATH=C:\jdk1.8.0_102\bin;%PATH%
set JAVA_HOME=C:\jdk1.8.0_102
```

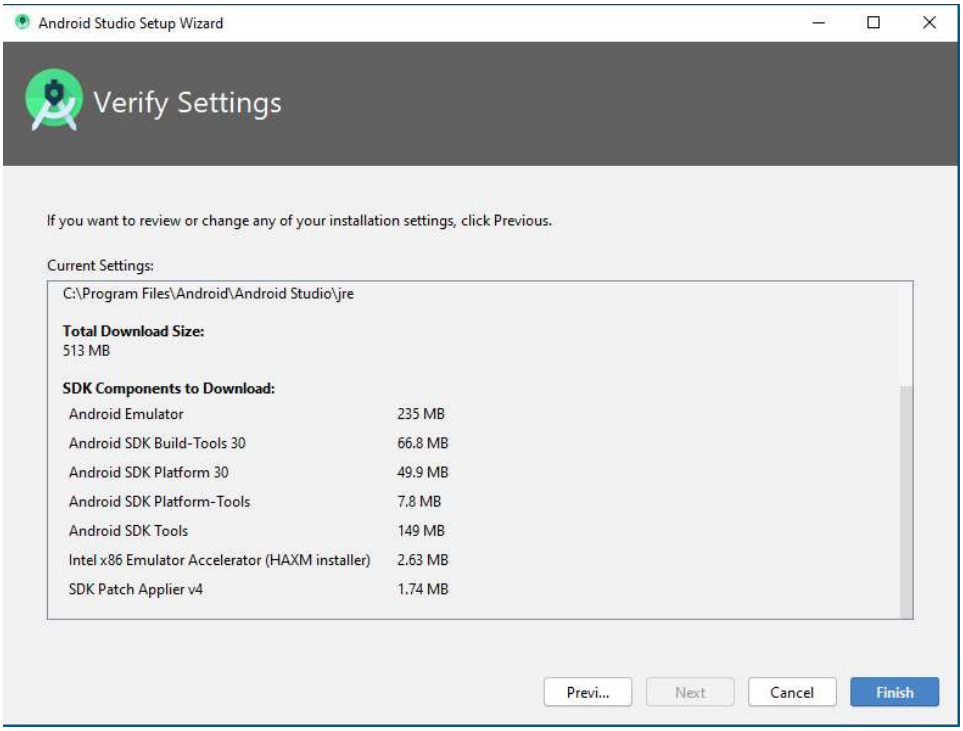
Alternatively, you could also right-click on My Computer, select Properties, then Advanced, then Environment Variables. Then, you would update the PATH value and press the OK button.

However, since we are going to use Android Studio, it will know automatically where you have installed your Java. Just choose the standard setup during your Android Studio installation.



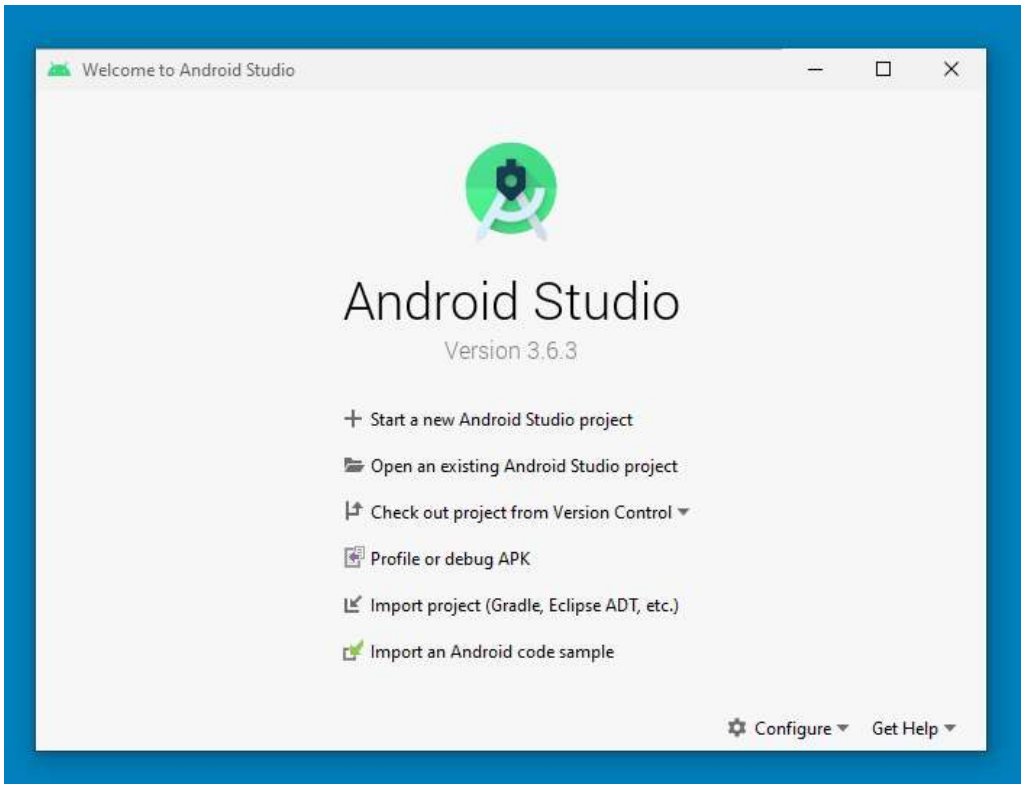
4. Download and install Android Studio from this link:
   https://developer.android.com/studio

5. Download Platforms and Packages.



If you chose the standard setup during your Android Studio installation, it will automatically download the recommended packages after you click **Finish**.

After successful downloads and installation, this should be the startup page of Android Studio.



SUPPLEMENTARY LEARNING RESOURCES

Android - Application components. (n.d.). RxJS, ggplot2, Python Data Persistence, Caffe2, PyBrain, Python Data Access, H2O, Colab, Theano, Flutter, KNime, Mean.js, Weka, Solidity. https://www.tutorialspoint.com/android/android_application_components.htm

Meet Android Studio. (n.d.). Android Developers. https://developer.android.com/studio/intro/

REFERENCES

Android - Application components. (n.d.). RxJS, ggplot2, Python Data Persistence, Caffe2, PyBrain, Python Data Access, H2O, Colab, Theano, Flutter, KNime, Mean.js, Weka, Solidity. https://www.tutorialspoint.com/android/android_application_components.htm

Android runtime (ART) and Dalvik. (n.d.). Android Open Source Project. https://source.android.com/devices/tech/dalvik

Android operating system: What you need to know. (n.d.). Investopedia. https://www.investopedia.com/terms/a/android-operating-system.asp

Characteristics of a successful mobile app. (2019, June 12). Seamgen Blog. https://www.seamgen.com/blog/5-characteristics-of-a-successful-app/

Developing Mobile Applications with Android. (2015). Active Learning, Inc.

Doan, T. H. (2020, February 3). *Virtual machine in Android: Everything you need to know*. Medium. https://android.jlelse.eu/virtual-machine-in-android-everything-you-need-to-know-9ec695f7313b

Meet Android Studio. (n.d.). Android Developers. https://developer.android.com/studio/intro/

Panhale, M. (2015). *Beginning hybrid mobile application development*. Apress.

Salmre, I. (2005). *Writing mobile code: Essential software engineering for building mobile applications*. Addison-Wesley Professional.

Understanding mobile apps. (2018, March 13). Consumer Information. https://www.consumer.ftc.gov/articles/0018-understanding-mobile-apps#basics