Lesson 1: Introduction to the Android Development Environment

INTRODUCTION

This lesson introduces the Android mobile marketplace and specific tools used to build mobile applications. The components of the Android Operating System and the development tools used to create Android applications are introduced. This lesson uses the Eclipse Integrated Development Environment (IDE) and Android Development Tools (ADT).

LESSON OBJECTIVES

By the end of this lesson, the student will be able to:

1. Identify the tools used to develop applications for Android Mobile devices.
2. Identify the basic components of the Android Operating System.
3. Identify the gestures common to mobile devices.
4. List the files used within an Android app.
5. Identify the components used with an app.
6. Demonstrate how to download and install Eclipse.
7. Demonstrate how to open the ADT and set up a workspace (Android Development Tools).
8. Demonstrate how to organize and start a new project using ADT (Android Development Tools).
9. Create the manifest file for an Android app using XML.

LEARNING SEQUENCE

Required Reading	Read the following: • Online Lesson Material
Resources	View the following: • Eclipse Android SDK and ADT download and install (11:55) • Android Application Development Tutorial – 5 – Overview of Project and Adding Folders (9:02) • Android Manifest file (5:58) Other Resources: • Android version history • Android Architecture • Behind the 'Internet of Things' Is Android—and It's Everywhere • Android developer website • Android design goals • Android Design Principles • Going Beyond Google Play in Distributing Your Android Apps • Eclipse website • Oracle website • Java Overview

	• Java-Basic Syntax • Dashboards
Assignments	Complete the following: 1. Lab: Creating Android Application Development Environment 2. Lab: Creating an Android Project, Device, and Your First App 3. Quiz

INSTRUCTION

The Android Platform

Android is an open source mobile operating system. Its source code is released by Google (which purchased Android, Inc. in 2005) under open source licenses. Since no company or individual directs the development of an open-source environment, organizations and developers can adapt and use the code for modifying the operating system or for creating any application (or app). Based on the Linux kernel, Android is designed to run on smartphones, tablets, and other touchscreen mobile devices. According to Wikipedia, there are 1 billion Android devices activated as of September, 2013.

The Android Operating System

The first version of Android for commercial use, Android 1.0, was released in September of 2008. Starting with releases in 2009, each version was alphabetically named based on a dessert or sweets theme. Visit Android version history to review the specific code names.

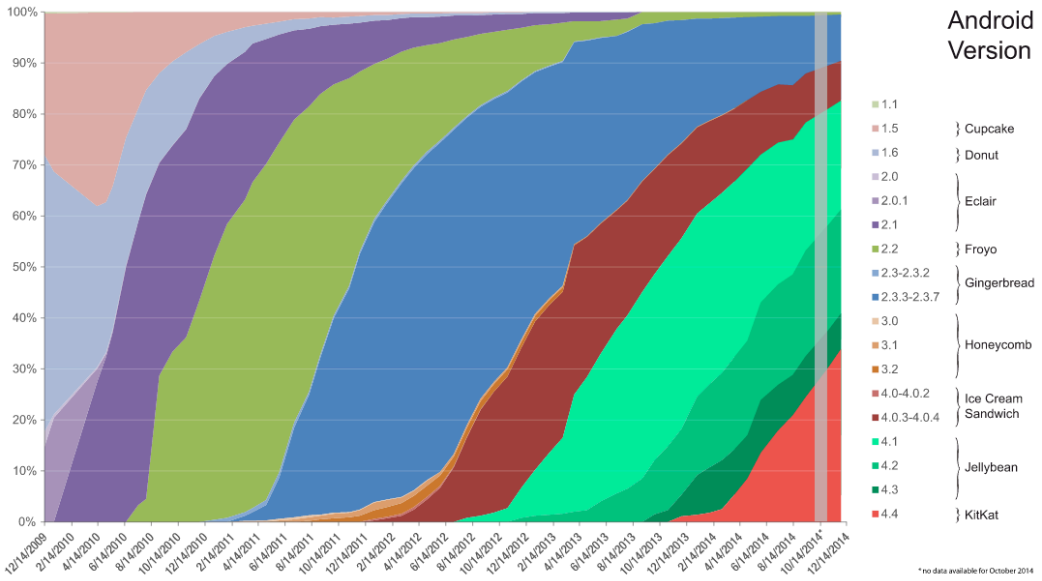


Figure 1 – Android version distribution

Figure 1 shows the distribution of each Android version starting in 2009 through 2014. The KitKat version of the operating system is currently the most popular among users.

Android Architecture takes a look at the software components that make up the Android operating system. All of the apps run at the top layer of the Android Architecture, the Applications layer. Keep this diagram in mind while progressing through the course.

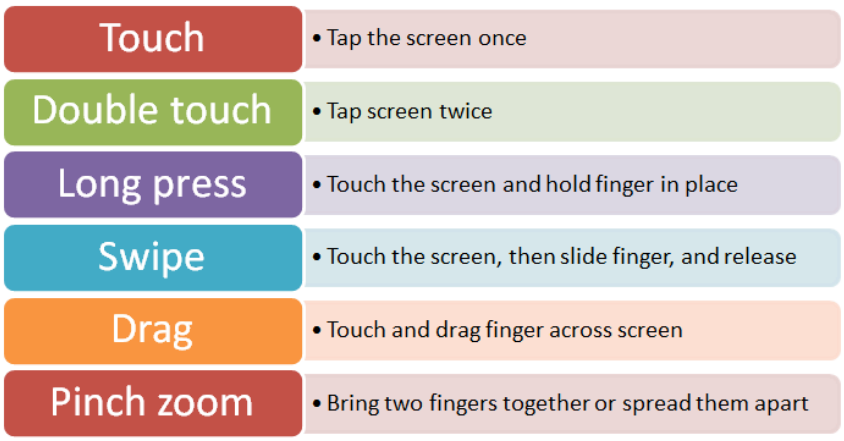
Android Features

Android devices include the obvious--smartphones, tablets, e-readers, and game consoles. Read the article, Behind the 'Internet of Things' Is Android—and It's Everywhere to learn about several other types of Android devices.

Android apps are written using the Java programming language. Java is an object-oriented programming language and has access to numerous class libraries to help develop apps quickly. Java is event driven, and the apps that a developer writes respond to the events and screen touches that the user of an Android device performs. By touching the screen, a user can navigate between running apps, play music, browse the web, or take a phone call.

The following gestures are common on the multi-touch screen of a mobile device.

Gestures on an Android device



App Considerations

There is more to app development than just the coding. There are also business considerations. For example, the demand for the app needs to be determined. Successful products are due to keen marketing strategies. There may also be financial decisions involved in the development.

Design

Before the coding even begins, a developer needs to identify the audience for the app. Next, set goals and look at the overall design. The Android developer website offers a lot of information on app design.

1. Go to http://developer.android.com.
2. Click on Design in the menu bar.

To create vision, the website lists the following goals:

- Enchant me
- Simplify my life
- Make me amazing

Click on the Design Principles link to read through the principles which guide the creative process along. Following sound interface design principles ensure that the app will make sense to the user.

Testing

Quality control needs to be a point of focus throughout the process. Testing the app at multiple steps along the way will keep any errors from getting buried or overlooked. Testing should not just be done at the end of development. After coding is complete, an app should undergo usability testing to ensure that users are not frustrated.

Once an app is published, feedback from users will help keep the app current and provide information to help keep the app updated after it appears in the Google Play Store. Be aware of any operating system updates because they can make an app obsolete. Apps will need to be updated at times when the operating system changes.

Get the Ball Rolling

After coming up with an idea for an app, go and see if it already exists in the Google Play Store. If there are already similar apps available, what would make the proposed app different? If there are no similar apps, ask why that is. Is there just no demand for that app, or is it because the proposed app is truly unique? Research the idea to determine if it is viable.

Expand on the design goals by looking at what is involved technically in creating the app. What are the hardware requirements? Look at the hardware that must be on the device in order for the app to work. Will there be any data and network requirements? For example, is the app pulling data from a database or perhaps writing to a database? What about going over a network?

Think ahead about how the app can be tested. Develop a strategy for handling bugs and keeping the app up-to-date. Maintaining the app and handling any problems that arise can require more time than the original development for the app to be successful.

The open source nature of Android combined with the established distribution channel and the volume of Android devices in the marketplace provide a great opportunity for anyone willing to take his or her programming knowledge one step further.

Distributing Android Apps

Android apps are sold and distributed through an online store called Google Play. Developers pay a one-time registration fee. After a developer registers, he or she can publish an app (assuming, of course, that the app meets the minimum standards). Updates to an app are also released through Google Play. If the developer is not going to charge for the app, Google Play publishes the app at no cost. If the developer wishes to charge for the app, the standard split is 70% to the developer and 30% to the wireless carrier.

Since the Android operating system can be modified by a manufacturer, a developer cannot assume that all devices have the Google Play store preinstalled. Device manufacturers like Amazon and Samsung, for example, have their own stores. The article Going Beyond Google Play in Distributing Your Android Apps discusses the other recommended App Stores that a developer should be familiar with.

Eclipse IDE

Android apps can be developed on a Windows, Linux, or Mac OS X system. This course uses Eclipse, an integrated development environment (IDE) that uses the latest Android Software Development Kit (SDK) and the latest Android Development Tools (ADT) plugin. The Eclipse IDE can be downloaded from the Eclipse website

Installing Eclipse is the first step in setting up the Android programming environment. The installation of Eclipse will also install the Android Software Development Kit (SDK), a set of development tools that can be used to design the app interface, write the code, and test the app. An Android emulator, included as part of the SDK, allows the developer to prototype and test Android apps without having access to a physical device.

Due to the number of different mobile devices available, there are a number of emulators within the Android SDK that enables a developer to target various devices and versions within the Android community. This also allows the app developer to test his app on a number of different devices without the need to actually possess the physical device.

Android Emulator

The emulator can simulate touching the screen by using the mouse and it can mimic all of the features of an Android device except for placing a phone call. There are a number of emulators available within the Android SDK to target the various devices and operating system versions available. Before running an emulator, an Android Virtual Device (AVD) must be created which defines the characteristics of the device such as screen size (in pixels), pixel density, the physical size of the screen, and size of the SD card for data storage. An app should be tested on several versions to make sure that the app is stable.

Android Developer Resources

Android is a complete framework which includes all of the tools needed to develop mobile apps. One place to start is the official Android developer website, the central location for all of the developer resources. There are links to the Software Development Kit (SDK), sample code, and videos in addition to guidelines for the design, development, and distribution of Android apps.

One of the system requirements is that Java Development Kit (JDK) 6 or higher is installed. JDK is not usually installed by default. It is also important to point out that having just the Java Runtime Environment (JRE) installed is not sufficient.

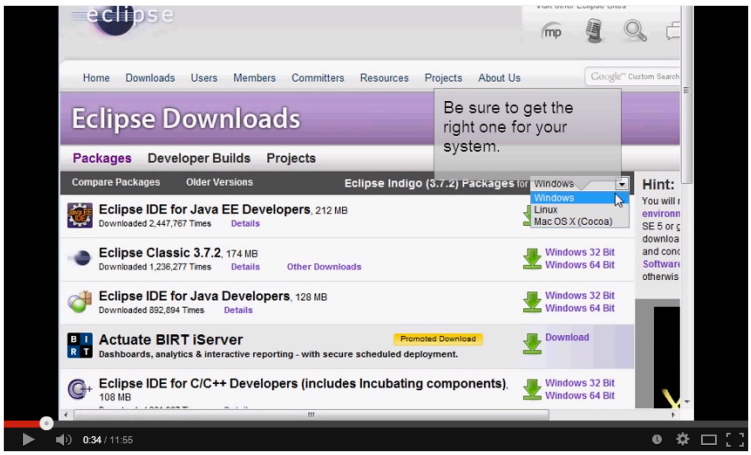
The JDK includes the Java Runtime Environment (JRE), the Java compiler, and the Java API.

- JRE is a basic virtual machine and is a required component to run any Java-based application.
- The Java compiler converts source code into Java bytecode, the instruction set of the java virtual machine.

- Java API describes the function of components which are pre-created and commonly used. The developer is able to use prewritten code via the Java API.

The JDK is available at the Oracle website.

Watch the video, Eclipse Android SDK and ADT download and install (11:55) for step-by-step instructions on installing Eclipse on a Windows system.



This video provides readers with an alternative to setting up the IDE on a Linux system used in the labs for this course.

Read through Java Overview for an introduction to the Java programming language. Java – Basic Syntax provides an overview of the components used in the Java programming language and has the reader create a simple Java program.

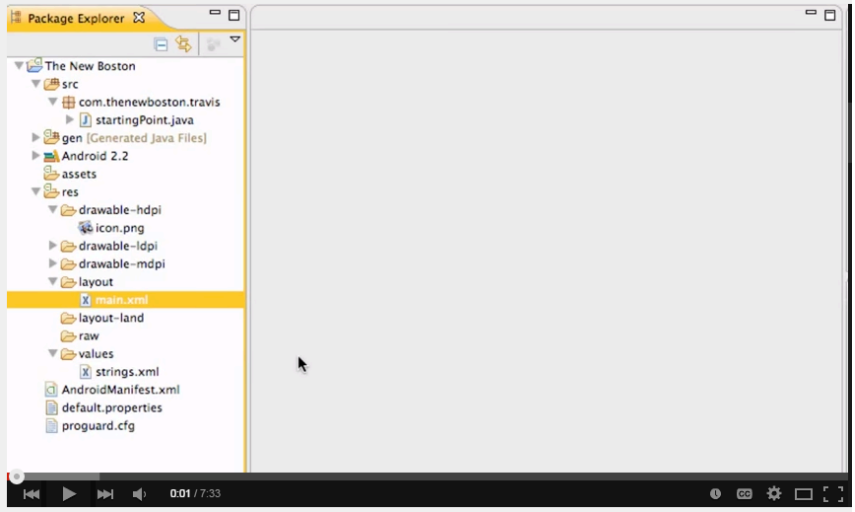
How Apps Work

Developers use the Java programming language to write Android apps. The SDK tools compile the code into an Android package (APK). The Android package file has an .apk file extension. The .apk file contains the entire app.

Each Android app has access to only the components that it requires to perform its function—each app has its own security sandbox. Think of an Android app as a user on a computer system. Both are assigned a unique user ID and both are assigned permissions for their own files. And just like on a computer system, it is possible to share resources.

Apps use two types of files—XML files and Java files. The XML files define what appears on the screen for the app's user interface. The Java files contain the logic for the app.

Watch the video, Android Application Development Tutorial – 5 – Overview of Project and Adding Folders (9:02) to take a tour of the folders and resources that are part of the Eclipse ADT.



App Components

There are four types of app components which make up the building blocks of an Android app:

- Activities occur on a single screen and provide a screen for user interaction. Each activity has a window with a user interface. One activity in an application is the "main activity". This means that it is the first activity the user sees when the application is launched.
- Services run in the background and do not provide a user interface. Services can continue to run in the background even when a user switches to another application. Playing music is an example of a service. The music will continue to play even if a user is playing a game.
- Content providers manage shared app data so that other apps can query or modify the data, if allowed. For example, a content provider is required to copy and paste complicated data from one application to another.
- Broadcast receivers respond to broadcast announcements (like if the battery is low, for example).

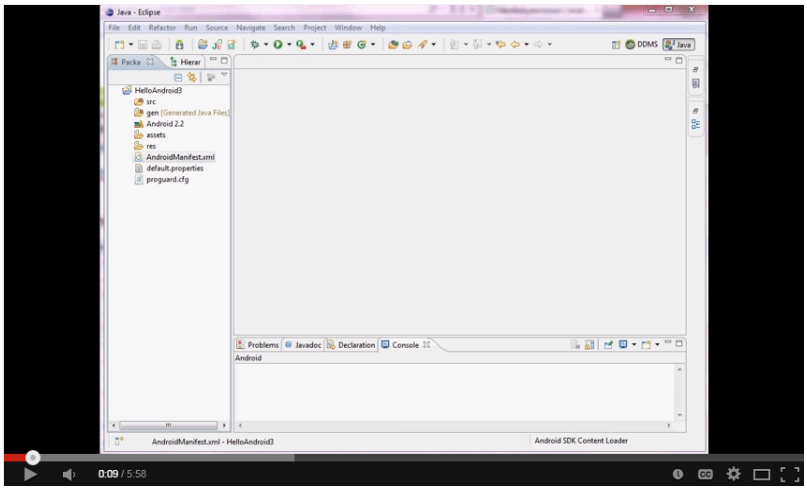
An intent is used to request an action from another app component. Intents can be used to start an activity, start a service or deliver a broadcast.

The Manifest File

One of the XML files used is the manifest file, AndroidManifest.xml. The manifest file is a required file and can be found in the root of the app project directory. The manifest file names the Java package for the app and all of the above components are declared here. This file:

- Identifies user permissions that the app requires
- Declares hardware and software features required
- Specifies API libraries that the app requires.

Watch the video, Android Manifest file (5:58) for a basic introduction to the manifest file using Eclipse.



Application Programming Interfaces (APIs)

Different devices may run different versions of the Android operating system. Refer back to Figure 1 to see a list of Android versions. Each version adds new application programming interfaces (APIs). An API level indicates which set of APIs are available. For example, Android 1.0 is API level 1. Android 4.4 is API level 19. Each Android version supports one API level. As parts of the API get upgraded, the new API is still compatible with earlier versions. The older replaced parts are deprecated rather than removed. Existing applications will still be able to use the replaced parts. Visit the Dashboards page on the Developers website to view up-to-date statistics on the number of devices running the available Android platforms.

App Resources

App resources include non-code resources such as images and audio files. Having these resources separate from the code allows a developer to provide alternative resources for different device configurations. For example, when a device's orientation is changed, the screen layout can also be changed.

Fundamental Concepts

The Android system allows a component in one app to activate a component in another app. Keep in mind the following fundamental concepts regarding the Android app framework:

- Apps provide multiple entry points which means that apps are built based on components that can be called individually. For example, an activity provides the screen for the user interface while a service can be working in the background.
- Different resources can be used for different device configurations.

Intents are used to start a second component from a first. For example, an intent is used by an activity to open a web page. Activities, services, and broadcast receivers are activated by an intent.

Example Manifest File

The manifest file in Figure 2 illustrates the concepts discussed in this lesson.

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
A    package="com.example.project1"
     android:versionCode="1"
     android:versionName="1.0" >

     <uses-sdk
B         android:minSdkVersion="8"
          android:targetSdkVersion="17" />

     <application
          android:allowBackup="true"
          android:icon="@drawable/ic_launcher"
          android:label="@string/app_name"
          android:theme="@style/AppTheme" >
          <activity android:name=".Logo"
C              android:label="@string/app_name" >
               <intent-filter>
                    <action android:name="android.intent.action.MAIN" />
                    <category android:name="android.intent.category.LAUNCHER" />
               </intent-filter>
          </activity>
          <activity android:name=".Menu"
               android:label="@string/app_name" >
               <intent-filter>
                    <action android:name="com.example.project1.MENU" />
                    <category android:name="android.intent.category.DEFAULT" />
               </intent-filter>
          </activity>
          <activity android:name="com.example.project1.MainActivity"
               android:label="@string/app_name" >
               <intent-filter>
                    <action android:name="android.intent.action.MAIN" />
                    <category android:name="android.intent.category.LAUNCHER" />
               </intent-filter>
          </activity>
          <activity android:name=".Accelerometer"
               android:label="@string/app_name" >
          </activity>
          <activity android:name=".ImageSwiper"
               android:label="@string/app_name" >
          </activity>
     </application>
</manifest>
```

Figure 2 – AndroidManifest.xml

Refer to Label A. The package name is defined within the <manifest> tag using the package attribute. The developer has the opportunity to version the app with two different version attributes as shown. The minimum API level that the app is compatible with is indicated as well as the target version.

Label B shows the <uses-sdk> tag required to define the minimum and targeted API Level that the app supports. The minimum API allows the Google Play store to use the information to filter which applications can be downloaded to specific devices. This helps to ensure that the app will work on the device. The <uses-sdk> tag is required for an app to be published on Google Play. Use the lowest API level possible for the app to be compatible with the most Android devices.

The targeted version should be the Android SDK version that the app was built and tested on.

Each activity within the app must be defined with an <activity> tag. Label C shows the activity class designated as the primary entry point for the app.

SUMMARY

While the mobile app marketplace continues to evolve, there are plenty of free and open-source resources and tools to use to develop skills. Employment opportunities for Android developers continue to grow. This lesson introduced the Android mobile marketplace and specific tools used to build mobile applications. The Eclipse IDE is one option for developing Android apps. The AndroidManifest.xml file, one of the two types of files used in an Android app was discussed.

ASSIGNMENTS

1. Lab: Creating Android Application Development Environment
2. Lab: Creating an Android Project, Device, and Your First App
3. Quiz