Mobile Architecture week 1 Chapter 1

1.1

Mobile operating system(mobile OS) - a device that controls the hard ware and the interaction between that hardware and software.

UI - user interface - How the user acts with the device

Apple apps are more strict than Google in being approved.

A native app is an app that is specific to the mobile OS and is created with the tools and programming languages supported by the mobile OS.

A web app is an app created with standard web technologies (HTML5, JavaScript, CSS, etc.) that runs in the mobile device's web browser .

A hybrid app is a native app uses a web container (like an internal web browser) to display the UI that is created with web technologies.

Android developers typically use the Android Studio IDE (Integrated Development Environment) and write Java or Kotlin code to produce Android native apps.

1.2

Three Identifiers for a android platform

1. A ***platform version*** that increases with each new Android platform release.
2. A ***code name*** that identifies the Android version with a dessert name (Ex: Donut), organized in alphabetical order.
3. An ***API level*** that identifies the Android Application Programming Interface (API) with an increasing integer.

Apps are usally desined for older models

Rooting is the choice to keep an older version of Android on your device

Android apps are written in Java or Kotlin using Android libraries from the Android SDK (Software Development Kit), which is a collection of libraries and tools for writing Android apps.

Process of Kotlin

An Android program written in Java or Kotlin is compiled into Java bytecode by a Java compiler. Then the R8 compiler from the Android SDK converts the Java bytecode into smaller, optimized Dalvik executable (Dex) bytecode. When an Android app is installed on a device, the Dex bytecode is converted into .oat files using an ahead-of-time (AOT) compiler. The .oat files are in a format that can be executed quicker than Dex bytecode. The Android Runtime (ART) is a virtual machine that runs the .oat files when the app is started. Versions of Android prior to Lollipop don't have ART; older Android versions use a Dalvik virtual machine to run Dex bytecode.

Apps are composed of 4 componets

Activity - a single screen and interaction with the user

Service - runs in the backgound a lenghty operation

Broadcast receiver - delievers system-wide broadcast announcments to the app

content Provider - manages an app's data that is stored on the file system, in a database, or on the web. Other apps use an app's content provider to access or manipulate the app's data

1.3

Android Studio is the official IDE for creating Android apps

The Android SDK and OpenJDK is installed with Android Studio.

OpenJDK is an open source implementation of the Java platform that includes a Java compiler.

Project - files containing: Apperacnce, behavior, and build configuration

Gradle build tool - takes project source files and combines the project files into a single APK file.

Android app Package(APK) - a collection of files for an app. noted with an .apk file extenstion.

Good practice is to avoid including passwords or other highly sensitive information in the Java source code.

Code obfuscation - is the process of renaming classes, methods, and variables and manipulating the code in other ways to obscure the code's meaning

* ***AndroidManifest.xml*** is an XML file that specifies important information about the app including the app name, theme, and permissions needed (like access to the camera).
* The ***java folder*** holds the app's Java source code and files to test the code. Java files have a .java file extension. MainActivity.java contains the code for MainActivity, the first activity that runs when the app is started.
* The ***res folder*** holds the app's resources. An ***app resource*** is any additional file or content that an app needs, including UI layouts, strings displayed in the UI, animation instructions, images, and audio files.
* The ***drawable folder*** holds the images that are displayed in the app.
* The ***layout folder*** holds the XML layout files. A ***layout*** defines an app's UI, which includes the visual structure and UI components that the user sees and interacts with. activity\_main.xml is the layout file for MainActivity.
* The ***mipmap folder*** holds the app's launcher icons. A ***launcher icon*** is an image that represents an app and is displayed on Android device's Home screen. The icon images are saved at different resolutions so the icons can display correctly on Android devices with different screen pixel densities.
* The ***values folder*** holds the XML files that provide various values used by the app like colors, strings, and styles.
* ***Gradle Scripts*** holds the app's configurable build files. The ***build.gradle*** file marked "Module: app" specifies app characteristics like targeted Android platform and library dependencies.

1.4 – Pizza Party App

1.5

App resources are located in various files in the Android project's res/ directory.

Resources - drawables, layouts, strings, and colors

A resource ID is composed of a resource type and resource name.

Ex: The resource ID layout.activity\_main has the resource type "layout" and resource name "activity\_main".

R.java - file containing an R class with subclasses named after each resource type and a static integer variable for each resource name

1.6

A value resource is a resource type stored in res/values/ that contains simple values like strings, integers, and colors.

A string resource is a value resource commonly defined in res/values/strings.xml. A string resource is created with the <string> elelement

The getString(resourceId) Activity method returns a String for the given resource ID.

The getText(resourceId) Activity method returns a CharSequence for the given resource ID that maintains HTML formatting.

Screen density is the number of pixels that fit within a fixed area of the screen and is usually referred to as DPI (dots per inch), where 1 dot is equivalent to 1 pixel. - more the clearer the image

A dimension resources - specify the width height, radius, size, or any other dimensions

* A ***dp*** (***density-independent pixel***) is an abstract unit based on the pixel density of the screen. dp units allow a view to maintain the same physical size on screens with different screen densities.
* A ***sp*** (***scale-independent pixel***) is an abstract unit based on the pixel density of the screen that is scaled by the user's font preference.
* A ***pt*** (point) is 1/72 of an inch on a screen, but the actual length depends on the screen density.
* A ***px*** (pixel) is an actual screen pixel. The number of pixels varies across devices and depends on screen density.
* A ***mm*** (millimeter) is a millimeter of a device's physical screen.
* A ***in*** (inch) is an inch of a device's physical screen.

Good practice is to use sp units for font sizes and dp units for all other dimensions to create UIs that look consistent on a variety of screen densities and font sizes that adjust to user preferences.

1.7

bitmap file - picture(GIF)

Portable Network Graphics (PNG) format compresses image information into a smaller file

***alternative drawables***, which are bitmap files saved at different sizes for different screen densities. The bitmaps are resized using medium density as the baseline:

* 0.75x for low density (ldpi)
* 1x for medium density (mdpi)
* 1.5x for high density (hdpi)
* 2x for extra-high density (xhdpi)
* 3x for extra-extra-high density (xxhdpi)
* 4x for extra-extra-extra-high density (xxxhdpi)

1.8 – debugging apps

System Log Codes

* ***Log.d(tag, message)*** sends a debug message to the system log. Debug messages help discover the cause of a bug. Ex: Logging the value of a variable is an appropriate debug message.
* ***Log.e(tag, message)*** sends an error message to the system log. Error messages report unrecoverable application errors. Ex: Logging that an unexpected exception is thrown is an appropriate error message.
* ***Log.i(tag, message)*** sends an information message to the system log. Information messages convey information about the app's progress or current status. Ex: Logging that a server connection is successful is an appropriate information message.
* ***Log.v(tag, message)*** sends a verbose message to the system log. Verbose messages are for all other types of messages. Ex: Logging the activity name that is currently running is an appropriate verbose message.
* ***Log.w(tag, message)*** sends a warning message to the system log. Warning messages warn about potential application problems. Ex: Logging that a file is getting close to a maximum limit is an appropriate warning message.

LOGCAT – is used to display system errors

If an Android app crashes because an uncaught exception is thrown in the code, the Logcat displays the exception's stack trace to aid developers in determining why the app crashed.

Debugger controls

* **F7** - Step into advances to a method's first line of code when calling a method
* **F8** - Step over advances to the next line of code without entering a method call
* **Shift+F8** - Step out advances to the next line of code outside the current method
* **F9** - Resume running the program

***Lint –*** a tool that identifies structural problems in an app's code and suggests fixes

1.9

Feature - any hardware or software divice that the app may wish to use and is able to.

Sandbox - a secuirty mechanism the aprotects the operating system and prevents apps from accessing restricted resources or infromation

Protection level

Normal - little risk

Dangersous - wants private information

XML Files – codes

* The ***<manifest>*** element's package attribute specifies the app's package name. Package names uniquely identify an app and cannot be changed once an app is published in Google Play Store.
* The ***<application>*** element uses various attributes to specify the app's icon file (android:icon), name (android:label), theme (android:theme), etc. The values point to app resources like @string/app\_name that can be modified to alter the app's name.
* The ***<activity>*** element specifies an app's activity name. An app may consist of multiple activities.
* The ***<intent-filter>*** element uses ***<action>*** set to android.intent.action.MAIN and ***<category>*** set to android.intent.category.LAUNCHER to indicate which activity is started first when the app is launched.
* Figure 1.9.1: AndroidManifest.xml used in Pizza Party app.
* <?**xml** version="1.0" encoding="utf-8"?>
* <manifest xmlns:android="http:**//**schemas.android.com/apk/res/android"
* package="com.zybooks.pizzaparty">
* <application
* android:allowBackup="true"
* android:icon="@mipmap/ic\_launcher"
* android:label="@string/app\_name"
* android:supportsRtl="true"
* android:theme="@style/AppTheme">
* <activity android:name=".MainActivity">
* <intent-filter>
* <action android:name="android.intent.action.MAIN" />
* <category android:name="android.intent.category.LAUNCHER" />
* </intent-filter>
* </activity>
* </application>
* </manifest>

1.10

***Model-View-Controller*** (***MVC***) is a popular architecture for developing applications with a user interface (UI). MVC organizes an app's code into three parts.

* The ***model*** contains the app's "business logic", the code that manages the data and logic of the application independent of the UI. In an Android app, the model is a collection of classes that the app's activities interact with.
* The ***view*** is responsible for displaying the app's UI and interacting with the user. In an Android app, the view is composed of the View objects that make up the UI.
* The ***controller*** contains the "application logic", the code that responds to the view and controls the flow of data between the model and view. In an Android app, the controller is often the activity code.

1.11

A unit test - tests part of a code to make sure it functions correctly

Legend of a unit test

@Test - Marks a public method as a test case to be executed.

@Before - Marks a public method that is executed before each test case is executed.

@After - Marks a public method that is executed after each test case is executed.

Table 1.11.1: Common JUnit assert methods.

|  |  |  |
| --- | --- | --- |
| Method | Description | Example |
| assertTrue()  assertFalse() | Verifies the given condition is true or false. | assertTrue(1 < 2);  assertFalse(earth.isFlat()); |
| assertEquals()  assertNotEquals() | Verifies two items are or are not equal. | assertEquals(greeting, "hello");  assertNotEquals(apples, oranges); |
| assertArrayEquals() | Verifies two arrays are equal. | assertArrayEquals(array1, array2); |
| assertNull()  assertNotNull() | Verifies the argument is or is not null. | assertNull(obj);  assertNotNull(obj); |
| assertSame()  assertNotSame() | Verifies two objects refer to the same or not the same object. | assertSame(obj1, obj2);  assertNotSame(obj1, obj2); |
| assertThat() | Verifies a value matches some matcher statement. | assertThat(num, is(5));  assertThat(str,  containsString("frog")); |

1.12

1. A ***local unit test*** is a test that has no dependencies on Android and is executed on a developer's machine. Ex: A sortArray() method requires nothing specific to Android to sort an array, so a local unit test is ideal for testing sortArray().
2. An ***instrumented unit test*** is a test that has dependencies on Android and is executed on an emulator or physical device. Ex: A loadColors() method may need access to a Context to load color resources, so testing loadColors() may require an instrumented unit test

A ***mock object*** is a temporary replacement for a real object.

This produces questionable results

1.13

Three types of tests

1. A ***UI test*** runs the app on an emulator or physical device and clicks on items, types text, etc. in the UI and verifies the UI displays the correct result. Ex: A UI test for a messaging app types a message, clicks the Send button, and verifies the app shows the message was sent.
2. An ***integration test*** verifies that multiple app components interact correctly. Integration tests run on an emulator or physical device and may interact with the UI. Ex: An integration test for a messaging app can verify the app's messaging service works correctly.
3. A ***unit test*** runs on a development machine or emulator/device and tests specific classes and methods. Ex: A unit test for a messaging app can verify the sendMessage() method returns false when the username parameter is null.

* *A****large test****is a UI test that executes longer than 1 second.*
* *A****medium test****is an integration test that executes in less than 1 second.*
* *A****small test****is a unit test that executes in 200 ms or less.*

***Test-drive development*** (***TDD***) is a software development process that requires developers to write a test first, then write just enough code to pass the test. TDD is often illustrated with the ***red-green-refactor*** cycle.