Introduction to Mobile Applications - 1

CSCI 4140: Open-Source Software Project Development

Prof. Wei Meng

https://course.cse.cuhk.edu.hk/~csci4140/

Based on slides by Mark Sherriff

Mobile Device Architecture

The Evolution of Mobile Devices

- Changes in phone design
  - Bag phones
  - Early handsets
  - Clamshell phones
  - Text & Internet
  - Keyboard-less smartphones
  - Bezel-less smartphones

The First Mobile Phone

- The Motorola DynaTAC 8000X
  - 1983
  - 13 x 1.75 x 3.5
  - 2.5 pounds
  - USD 3,995
  - +Monthly fee
  - Pay per minute

The Evolution of Mobile Devices (Cont.)

- Mobile phones had been just phones for a while
- Mobile phones stopped being a novelty
- Batteries got better, coverage got better, plans got better, displays got better, …
- The need for browsing content on the Internet rose
- The first mobile web platform was born

Wireless Application Protocol (WAP)

- A technical standard for accessing information over a mobile wireless network
- Basically a protocol stack allowing WAP equipment and software communicate over different mobile network technologies (e.g., GSM, CDMA)
- Initially, Wireless Markup Language (WML) was used instead of HTML
- WAP achieved popularity in early 2000s

The Evolution of Mobile Devices (Cont.)

- In early 2000s, purchases were made through SMS
  - Sending a text message to a pay-per-text number to download wallpaper or ringtone, etc.
- The Internet has been full of media that people want to consume on the go
- Mobile phones seemed like an obvious next step
  - A device that everyone carries and is always connected

Bigger Players Got Involved

- Nokia was dominating on the mobile phone market early on
- Other players (e.g., Blackberry, Samsung, HTC) were also involved
- Each manufacture had their own operating system, which made developing third-party apps difficult
- Transmission speeds of mobile networks got improved
- Phones started running known OSs (Windows CE and Linux)
- Handset manufactures decided to open up

Fractured Mobile Market

- Microsoft
  - “Write once, run on any Windows device”
  - Worked for a while with PDAs
- Apple
  - Started with a phone that had a web browser and apps - iPhone
  - Evolved into much more with the App Store
- Google
  - Just provided the OS and let others build the hardware (for a while)
  - Open source OS + no developer fee => lots of interest and apps

Two Main Mobile OSs Remain

- iOS
  - Apple mobile devices only, e.g., iPhone, iPad, iPod
  - Objective-C or Swift using Xcode
  - Not open
- Android
  - Thousands of devices with diverse specifications
  - Java or Kotlin using Android Studio
  - Open source

Android

The Basics

- All apps are written in Kotlin or Java
    - This is not necessarily the case
- An Android application is compiled/packed into an Android Package (APK) file
    - APK files a type of archive file in zip format
- APKs must be digitally signed with a certificate before they can be installed
    - You can sign the APK with a debug certificate for debugging

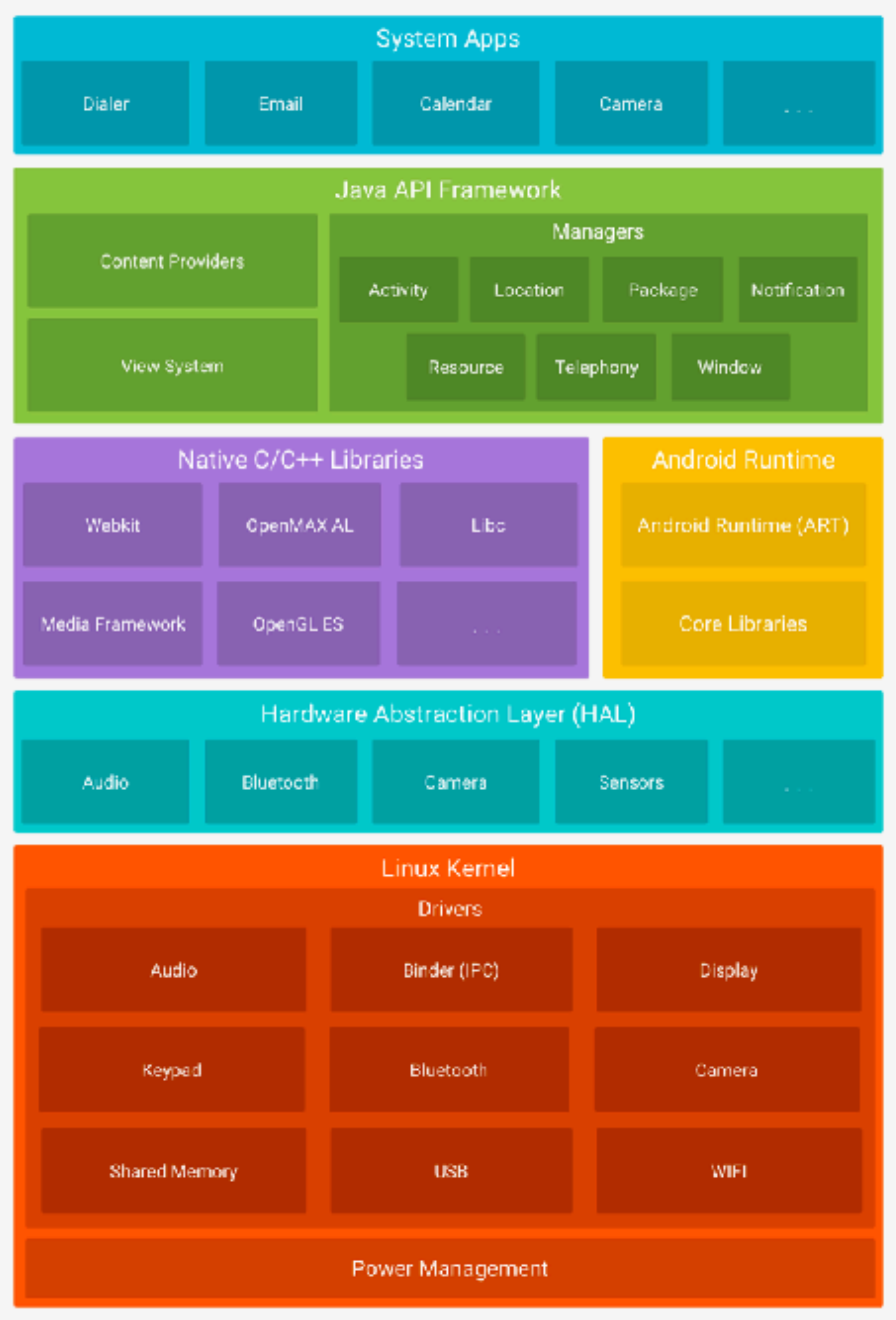
The Basics (Cont.)

- Android is a multi-user OS (Linux)
- At installation time, Android gives each package a distinct Linux user ID
- Each Android application executes in a sandbox, which isolates the app data and code execution from other app
  - Dalvik was Android’s runtime virtual machine
  - Sandbox is implemented at the OS level
  - How do apps share resources and data?

Sharing Data Across Apps

- Two Android apps can have the same Linux user ID
  - Setting *sharedUserLabel* and *sharedUserId* in the AndroidManifest.xml file
- Intent
- Content Provider / File Provider
- Shared storage

Platform Architecture



System Apps

Dialer, Email, Calendar, Camera, . . .

Java API Framework

Content Providers

View System

Managers

Activity, Location, Package, Notification, Resource, Telephony, Window

Native C/C++ Libraries

Webkit, OpenMAX AL, Libc, Media Framework, OpenGL ES, . . .

Android Runtime

Android Runtime (ART)

Core Libraries

Hardware Abstraction Layer (HAL)

Audio, Bluetooth, Camera, Sensors, . . .

Linux Kernel

Drivers

Audio, Binder (IPC), Display, Keypad, Bluetooth, Camera, Shared Memory, USB, WIFI

Power Management

Runtime

- What is a runtime?
    - A system implementing portions of an execution model
    - The instructions/software that are executed while a program is running
    - For example, an interpreter is part of a runtime of a programming language

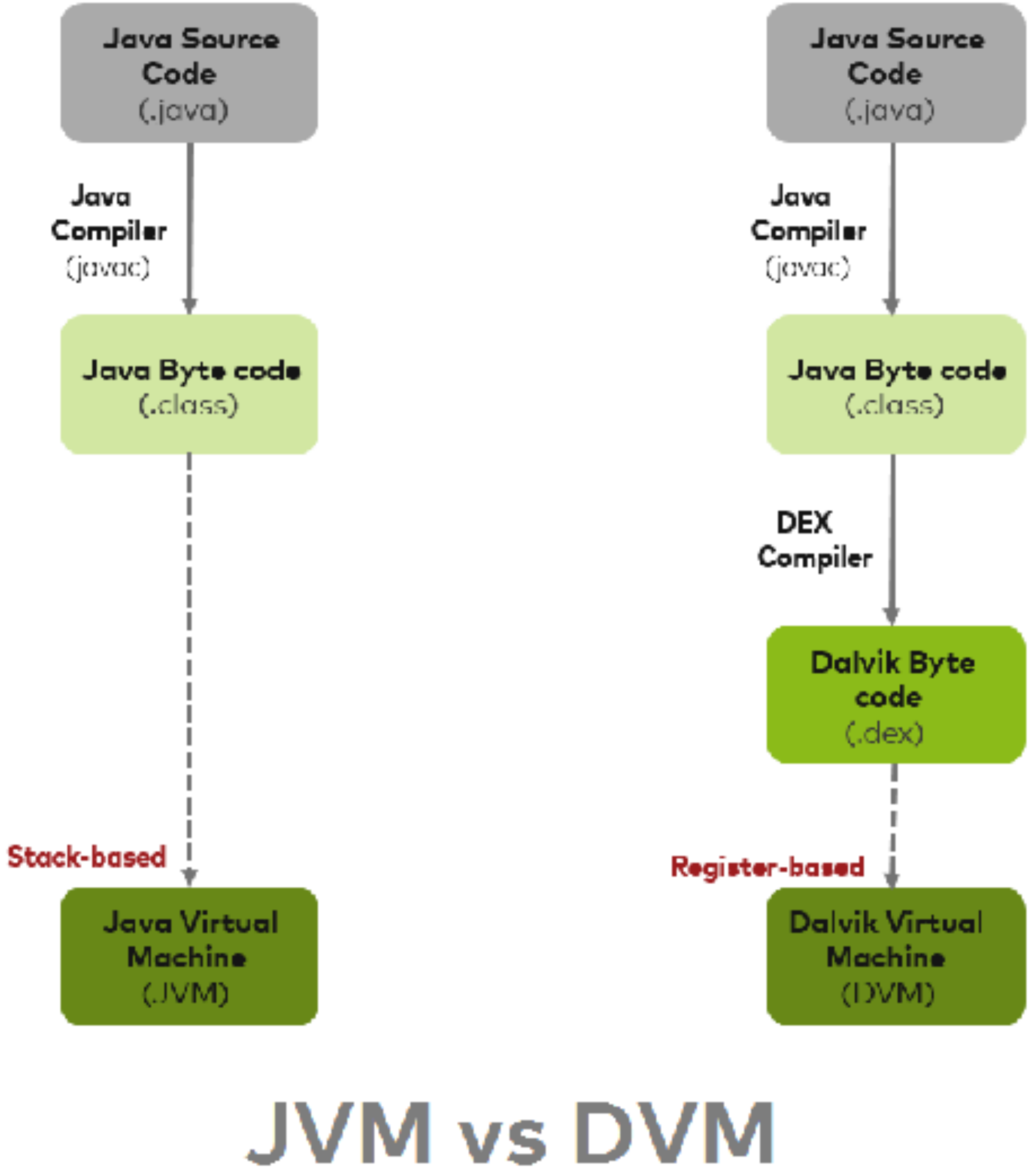
How A Java Program Run

- Java programs are executed in a runtime called Java Virtual Machine (JVM)
- The Java compiler translates Java code into byte code (.class files), i.e., an intermediary representation
- The JVM converts the byte code into machine code and executes the machine code

What is the benefit of using Virtual Machine?

Android Runtime

- Dalvik was Android’s runtime virtual machine



Dalvik uses Just-In-Time (JIT) compilation

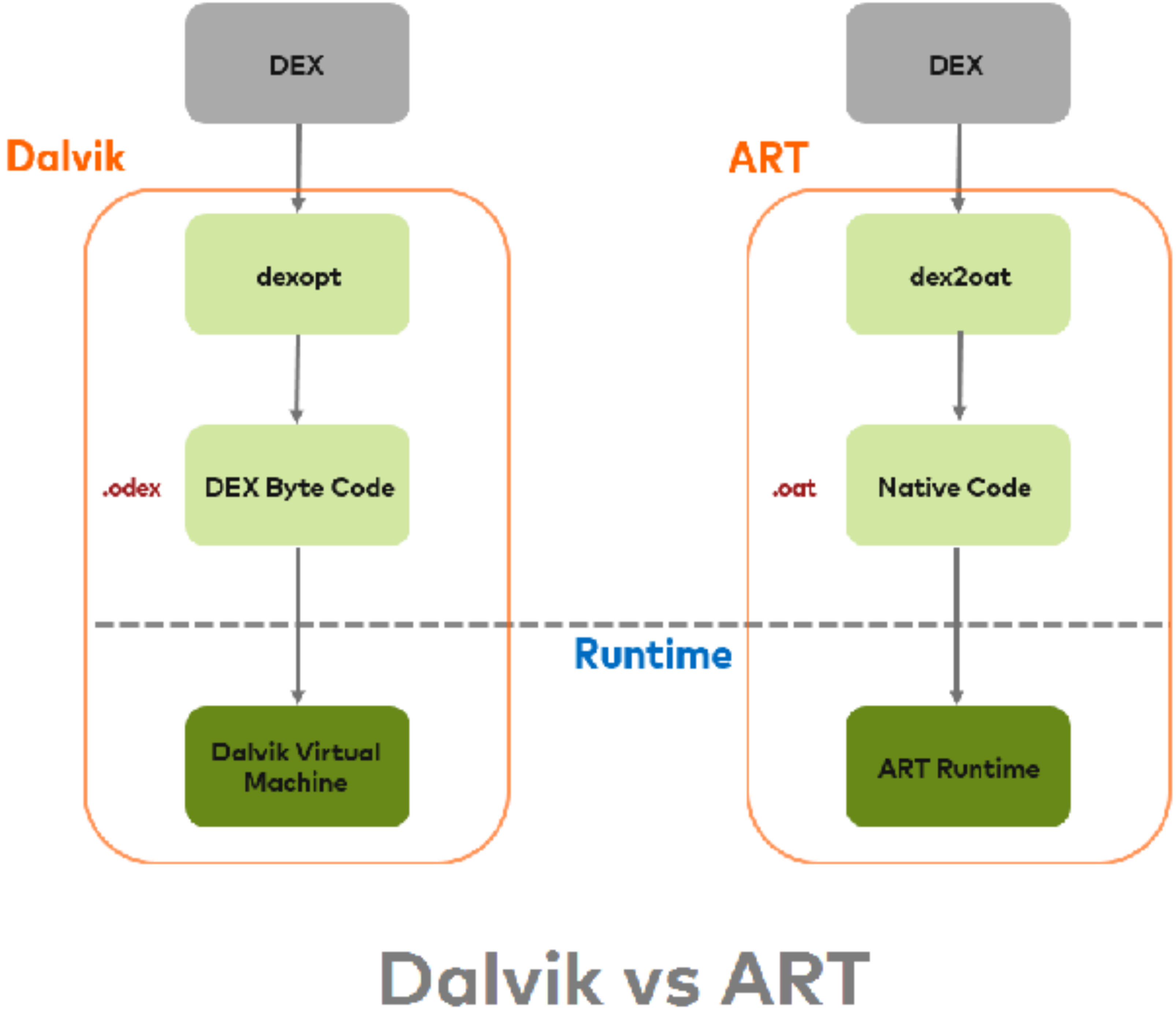
What is Just-In-Time?

JIT is also known as dynamic compilation. Computer code is compiled during the execution of a program.

Dalvik compiles a part of the byte code when an app runs.
More code is compiled and cached as the execution progresses.

Android Runtime (Cont.)

- ART (Android runtime) has replaced Dalvik



ART uses Ahead-of-Time (AOT) compilation

ART and Dalvik are compatible runtimes running Dex byte code

ART compiles the Dex byte code of an app at its installation time

Android Runtime (Cont.)

- Benefits of ART
  - Apps run/start faster
  - Apps consume less energy, why?
  - Improved garbage collector and developer tools
- Drawbacks of ART
  - Installation takes more time
  - Apps require more device space

Main Components

- Activities – represent a single screen with a UI
- Services – represent tasks running in the background
- Broadcast Receiver – listens for system-wide messages to respond to
- Content Provider – manage access to data of apps
- Intent – an abstract description of an operation to be performed and the glue between activities
- Application – a set of Activities that make up a cohesive unit

Activity

- Conceptually, an Activity is a single screen of your application
- It is the entry point for interacting with the user
- In other words, an app is a collection of activities
- An activity represents both a screen and a feature
    - Login activity, main activity, etc.
- Apps can start activities in other apps

Service

- A Service is a component that runs in the background to perform long-running operations or to perform work for remote processes

- A service does not provide a user interface

- A service is a general-purpose entry point for keeping an app running in the background

- Examples of Services:
  - Playing music in background
  - Fetching data from remote servers

Broadcast Receivers

- A Broadcast Receiver responds to system-wide broadcast announcements
- It enables the system to deliver events to the app outside of a regular user flow
  - The apps don’t need to be running
- Many broadcasts originate from the system
  - The screen has turned off, the battery is low, or a picture was captured
- Apps can also initiate broadcasts
- Broadcast Receivers typically don’t have a UI, but could create a status bar notification

Content Provider

- A Content Provider manages a shared set of app data
- This shared set of data could be a file, an SQLite DB, a remote link to a web service, etc.
- Other apps can query or modify the data through the Content Provider if they have the permission
  - Android system has a content provider for managing the user’s contact information
- Content providers are also useful for reading and writing private app data

Intent

- An Intent is a message that requests an action from another component of the system
  - This includes the “please start up your App” Intent that the system sends when a user clicks on your App icon
- Due to the component nature of Apps (made up of Activities, Services, etc.), it is easy to build features of your App using existing system components
  - For example, an app can request the Camera app to capture a photo instead of writing code to implement the camera feature
- An Intent is delivered to the system. The system then activates the component in other apps.

Where Is the UI?

- The User Interface for an Android App is defined in the layout xml files

- Each layout xml file should correspond to an Activity

App Resources

- Resources are the additional files and static content that your code uses
    - Layout definitions
    - Images
    - User interface strings
    - Animation instructions
    - …

App Manifest

- Every app project must have an AndroidManifest.xml file (with precisely that name)
- The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Play
- The following needs to be declared in the manifest
  - App's package name
  - Components (Activities, Services, Broadcast receivers and Content Providers)
  - Permissions
  - Specific hardware and software features the app depends on

App Permissions

- Apps have to request permission before they can use certain system data and features

- Android may grant the permission automatically

- Some sensitive permissions need to be explicitly approved by a user at installation time or execution time

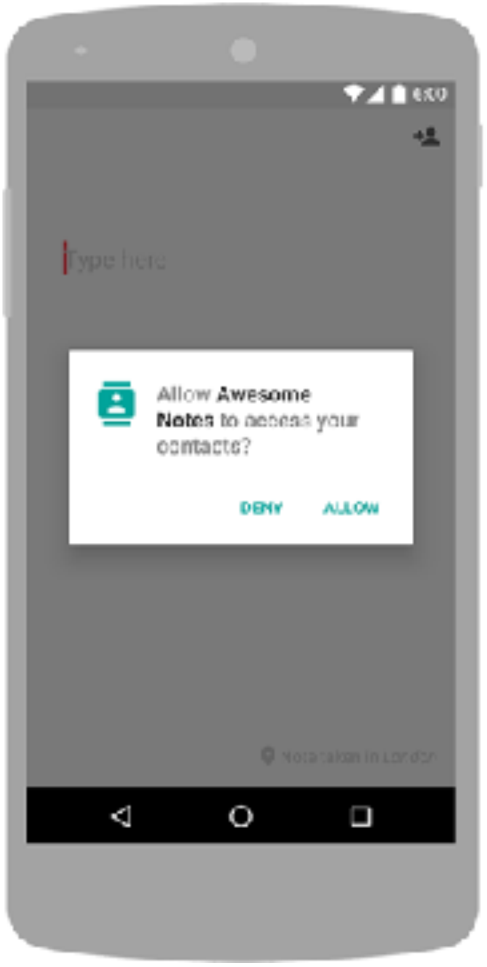
- The purpose of a permission is to protect the privacy of an Android user

Permission Approval

- An app must declare its required permission using <uses-permission> tags in the app manifest

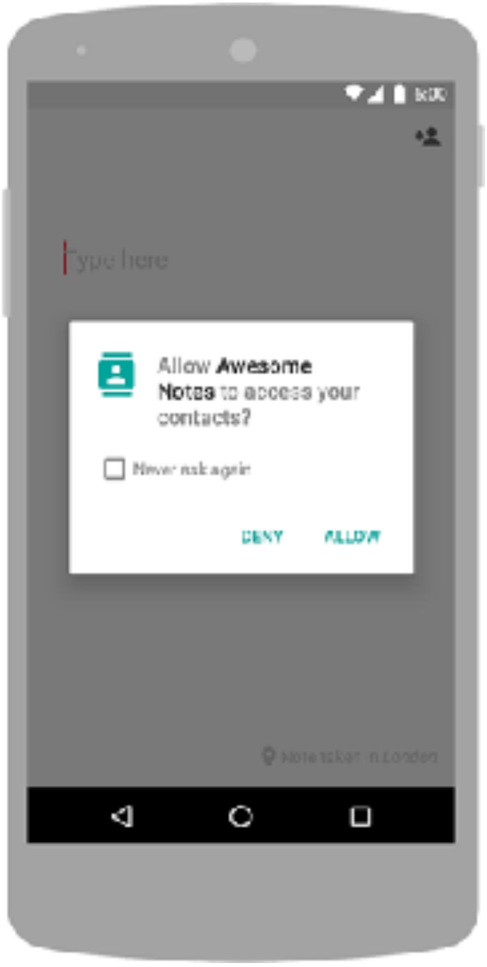
- The system (6.0+) generates request prompts for dangerous permissions, e.g., SEND_SMS

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```



Type here

Allow Awesome Notes to access your contacts?



Type here

Allow Awesome Notes to access your contacts?

☐ Never ask again