


# CSCE 315: Android Lectures (1/2)

- Dr. Jaerock Kwon
- Kettering University

1





## App Development for Mobile Devices

Jaerock Kwon, Ph.D. Assistant Professor in Computer Engineering

Announcement

2



## Lecture 1

### Introduction to Android

Kettering University

# Today's Topics

4

- Android Introduction
- Java crash course

# Android

5

# What is Android?

6

- An open source **software stack** that includes
  - Operating system
    - Linux operating system kernel that provides low level interface with the hardware, memory management, and process control.
  - Middleware
    - A run time to execute Android applications including Dalvik virtual machine and core libraries.
  - Key mobile applications
    - Email, SMS, PIM, web browser, and etc.
  - Along with API libraries for writing mobile applications.
    - Including open-source libraries such as SQLite, WebKit, and OpenGL ES.
- Open-source development platform for creating mobile applications.

# Android

7

- Complete
  - A complete set of software for mobile devices: an OS, middleware, and key mobile apps.
- Open
  - It was built to truly open.
- Equal
  - All apps are created equal.
  - No different between the phone's core apps and third-party apps.
  - Equal access to a phone's capabilities.
- Breaking down app boundaries
- Fast & easy app development



# Android SDK Features

8

- No licensing, distributions, or development fees or release approval processes.
- GSM, EDGE, and 3G networks for telephony and data transfer
- Full multimedia hardware control
- APIs for using sensor hardware including accelerometer and the compass.
- APIs for location based services
- IPC
- Shared data storage
- Background applications and processes.
- Home screen widgets, Live Folders.
- HTML5 WebKit-based web browser
- And many more...

# Introducing the Development Framework

9

# Android SDK

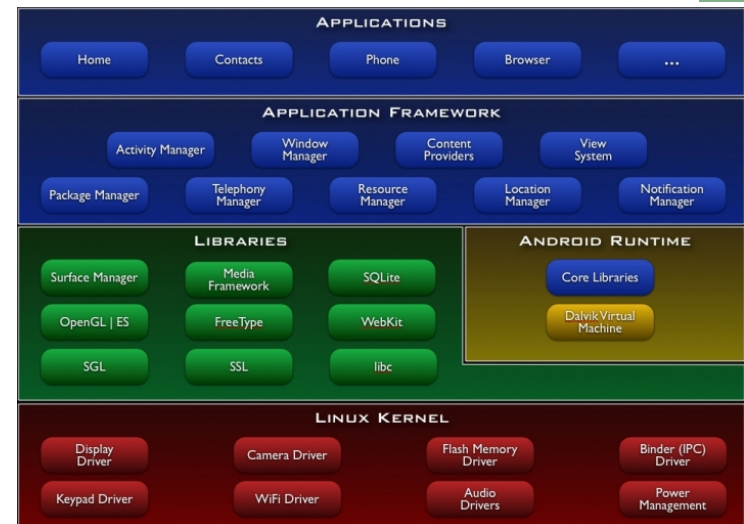
10

- The Android SDK includes
  - The Android APIs
    - The core of the SDK
  - Development tools
    - These tools let you compile and debug your app.
  - The Android Virtual Device Manager and Emulator
    - Android Emulator:
      - You can see how your applications will look and behavior on a real Android device
      - All Android applications run within Dalvik VM.
  - Documentations
  - Sample code
- No IDE from Google
  - There is no dedicated IDE for Android from Google.
  - Eclipse IDE:
    - Android has a special plug-in for Eclipse IDE (ADT Plugin for Eclipse) for creating Android projects.
    - ADT Plugin tightly integrates Eclipse with the Android Emulator and debugging tools.

# Android Software Stack

11

Android architecture



# Application Framework

12

- Android offers developers the ability to build rich and innovative applications.
- Developers have full access to the **same** framework APIs used by the core applications.
- Underlying all applications is a set of services, including
  - Views
    - can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser
  - Content Providers
    - enable applications to access data from other applications (such as Contacts), or to share their own data
  - A Resource Manager
    - provides access to non-code resources such as localized strings, graphics, and layout files
  - A Notification Manager
    - enables all applications to display custom alerts in the status bar
  - An Activity Manager
    - manages the lifecycle of applications and provides a common navigation backstack

# Libraries

13

- A set of C/C++ libraries used by various components of the Android system.
  - System C library
    - Tuned for embedded Linux-based devices
  - Media Libraries
    - Based on PacketVideo's OpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files
  - Surface Manager
    - Manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
  - WebKit
    - A modern web browser engine which powers both the Android browser and an embeddable web view
  - SGL/ 3D libraries
    - SGL: underlying 2D graphics engine
    - An implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer
  - FreeType
    - bitmap and vector font rendering
  - SQLite
    - A powerful and lightweight relational database engine available to all applications

# Android Run-time

14

- Android includes a set of core libraries that most of the functionality available in the core libraries of the Java programming language.
- Every Android app runs in its own process with its own instance of the Dalvik virtual machine.
- The Dalvik VM executes files in the Dalvik Executable (.dex) format.

15

# Java Crash Course

# Java

16

- A programming language
  - Syntax is very similar to C++ but different!
- A virtual platform
  - Java virtual machine is a software machine or hypothetical chip.
    - Note: The Dalvik virtual machine in Android is optimized for small footprint machine.
  - Bytecodes (cross-platform binary code)
    - .class binary file of bytecodes
- A class libraries
  - APIs for GUI, data storage, I/O, networking, and etc.

# Java language

17

- No code outside of the class definition.
- Single inheritance only.
- Only one top level public class in a file
  - The file name should be same as the public class name.

# Java Bytecode & Virtual Machine

18

- Bytecode (the `class` file) is an intermediate representation of the program.
  - You can consider bytecode as the machine code of the Java Virtual Machine.
- Java interpreter starts up a new virtual machine when it runs a Java bytecode.

# Package and Reference

19

- Packages and import
  - A package is a bunch of classes and interfaces.
    - Library of classes
  - You can import packages that you need.
    - Example) `import android.os.Bundle`
- Reference
  - No pointers!
    - Java doesn't have pointer variables.
    - Reference variables are equivalent in concept.
  - Objects and Arrays are reference types
    - Primitive types are stored as values

# Creating Objects

20

- `Point p;`
  - Note for C++ programmer.
    - This doesn't create the object of Point class unlike in C/C++.
    - This is only declaration of a variable.
    - Remember there is no pointer in Java.
- `Point p = new Point(1, 2);`
  - This allocates an object.
- Garbage collector
  - It reclaims unused memory.
  - You don't need to free unused objects.

# Reference

21

- Example
  - `int x = 10;`  
`int y = x;`  
`// x has a separate memory space from y`
  - `Point p = new Point(1, 2);`  
`Point q = p;`  
`// q is a reference to p;`  
`// there is only one copy of Point object in the memory`

# Passing Arguments

22

- Primitive type:
  - Pass by value:
    - The called method has a copy of the value.
    - The method cannot pass changed value in the argument to the caller.
- Reference type:
  - Pass by reference:
    - The called method has a copy of the reference.
    - The method accesses the same object!

# Inheritance

23

- Keyword **extends** to inherit from a superclass.
- Example
  - `package edu.kettering.hellokettering;`  
  
`import android.app.Activity;`  
`import android.os.Bundle;`  
  
`public class HelloKettering extends Activity {`  
 `/** Called when the activity is first created. */`  
 `@Override`  
 `public void onCreate(Bundle savedInstanceState) {`  
 `super.onCreate(savedInstanceState);`  
 `setContentView(R.layout.main);`  
 `}`  
`}`

## Create a sample project

24

## Developing for Mobile Apps

25

## Design Considerations

26

- Small and portable mobile devices
  - Offer exciting opportunities for software development.
  - But consider limitations
    - Low processor power
    - Limited RAM/permanent storage capacity
    - Small screen size
    - High costs associated with data transfer
    - Slow data transfer rates with high latency
    - Unreliable data connections
    - Battery life!
- Designing for Android
  - Performance
  - Responsiveness
  - Seamlessness

## Designing for Performance

27

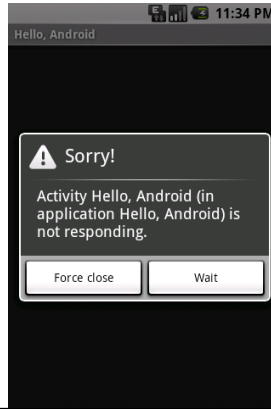
- Being fast and efficient
  - <http://developer.android.com/guide/practices/design/performance.html>
  - Avoid creating short-tem temporary objects.
    - Fewer objects created mean less-frequent garbage collection
  - Avoid internal getter/setters
    - Excellent habits for C++, but not for Android.
    - Direct field access is about 7x faster than invoking a trivial getter/setter.
  - Use `static final` for constants
  - Use `enhance` for loop syntax

# Designing for Responsiveness

28

## Application Not Responding (ANR)

- Activity Manager and Window Manager monitor application responsiveness.
- Android display the ANR dialog when it detects one of following conditions
  - No response to an input event within 5 seconds
  - A `BroadcastReceiver` hasn't finished executing within 10 seconds
- How to avoid ANR?
  - When an Android app runs on a single thread, any lengthy operation (network, database, computationally expensive calculation) could invoke the ANR.
  - Consider making a child thread to do the lengthy operation.



# Designing for Seamlessness

29

- Your application can cause problems under the multitasking environment when you ignore seamlessness issues.
- Be a good citizen!
  - Save instant state
    - Keep in mind that Android is a mobile platform.
    - Another app can pop up any time over your own app
  - Use a thread when you need to do a lot.
    - Avoid the ANR.
  - Use multiple screens when necessary.
  - Design your UI to work with multiple screen resolutions
  - Assume the network is slow
  - Don't assume touchscreen or keyboard
  - Do conserve the device battery

30

Questions?