Introduction

Programming the Android Platform
Logistics

- Professor
  - Adam Porter
  - aporter@cs.umd.edu
  - Office: AVW #4125
  - Office hours: M. 1-2pm & W. 10-11am
    - Or by appt.

- TA
  - Derek Juba
  - juba@cs.umd.edu
  - Office: TA Room
  - Office hours: W. 1-2pm & F. 2-3pm
Goals

- Learn about
  - Mobile devices
  - Mobile device programming
  - The Android platform
- Develop interesting Android applications
  - Expect lots of programming
  - Each student will do multiple projects

Yesterday, I was a guest on the Kojo Nnamdi Show (WAMU 88.5 FM) where I discussed the booming job market in mobile application development.
  - Tech Tuesday: Where the Jobs Are
Class Organization

- Mix of lecture and programming exercises
  - ½ presentation
  - 1/2 laboratory exercises & semester project
- Organization will remain flexible
  - Will change as needed
The Android Platform

- A software stack for mobile devices:
  - Operating system, middleware & key applications
- Use Android SDK to create applications
  - Libraries & development tools
- Lots of documentation
  - Start browsing today!
The Android Architecture

**Linux Kernel**

- Abstraction layer between HW & SW
- Provides services such as:
  - Security
  - Memory & process management
  - Network stack
  - Device driver model
Linux Kernel (cont.)

- Android-specific components
  - Binder – IPC
  - Android shared memory
  - Power management
  - Alarm driver
  - Low memory killer
  - Kernel debugger & Logger
Native Libraries

- **C/C++ libraries**
  - System C library
    - bionic libc
  - Surface Manager
    - display management
  - Media Framework
    - audio/video
  - Webkit
    - web browser engine
  - OpenGL ES, SGL
    - graphics engines
  - SQLite
    - relational database engine
  - SSL
    - secure sockets layer
Android Runtime

- Support services for executing applications
  - Core Libraries
  - Dalvik Virtual Machine
Core Libraries

- Core Java classes
  - android.*
  - java.*, javax.*
  - junit.*
  - org.apache.*, org.json.*, org.xml.*
- Doesn’t include all standard Java SDK classes
  - developer.android.com/reference/packages.html
  - www.zdnet.com/blog/burnette/java-vs-android-apis/504
Applications typically written in Java
- Do not run in a standard Java virtual machine
- `dx` program transforms java classes into .dex-formatted bytecodes
- Bytecodes executed in Dalvik Virtual Machine
- Applications typically run in their own processes, inside their own instance of the Dalvik VM
• Dalvik VM designed to run on a handset
  ▪ Slow CPU
  ▪ Little RAM
    ▪ e.g., 64Mb total, ~10Mb available at runtime
  ▪ No swap space
  ▪ Limited battery life
Dalvik Virtual Machine (cont.)

- Some Dalvik design choices
- One .dex file for multiple classes
- Modified garbage collection to improve memory sharing
- Optimizations applied at installation time
- Register-based, rather than stack-based
Dalvik Virtual Machine (cont.)

- Memory
  - .dex file has common constant pool for multiple classes
  - Modified garbage collection to improve memory sharing
- CPU
  - Optimizations at installation time
  - Register-based, rather than stack-based
Using Registers

- Expected benefits over stack-based VMs
  - Avoids slow instruction dispatch
  - Avoids unnecessary memory accesses
  - More efficient instruction stream
public static long sumArray(int[] arr) {
    long sum = 0;
    for (int i : arr) {
        sum += i;
    }
    return sum;
}
Java Bytecode

0: lconst_0
1: lstore_1
2: aload_0
3: astore_3
4: aload_3
5: arraylength
6: istore 4
8: iconst_0
9: istore 5
11: iload 5
13: iload 4
15: if_icmpge 36
18: aload_3
19: iload 5
21: iload
22: istore 6
24: lload_1
25: iload 6
27: i2l
28: ladd
29: lstore_1
30: iinc 5, 1
33: goto 11
36: lload_1
37: lreturn

% javap -c ClassName
0000: const-wide/16 v0, #long 0 // #0000
0002: array-length v2, v8
0003: const/4 v3, #int 0 // #0
0004: move v7, v3
0005: move-wide v3, v0
0006: move v0, v7
0007: if-ge v0, v2, 0010 // +0009
0009: aget v1, v8, v0
000b: int-to-long v5, v1
000c: add-long/2addr v3, v5
000d: add-int/lit8 v0, v0, #int 1 // #01
000f: goto 0007 // -0008
0010: return-wide v3

% dexdump –d classes.dex
Register-based vs stack-based VMs:

- 30% fewer instructions
- 35% fewer code units (1-byte vs. 2-byte instructions)
- 35% more bytes in the instruction stream
  - but can consume instructions two bytes at a time

* See www.youtube.com/watch?v=ptjedOZEXPM
Application Framework

- Window Manager
  - Manages top-level window’s look & behavior
- View system
  - lists, grids, text boxes, buttons, etc.
- Content Providers
  - Inter-application data sharing
- Activity Manager
  - Application lifecycle and common navigation stack
Application Framework (cont.)

- Package manager
  - Manages application packages
- Telephony manager
  - State of telephony services
- Resource Manager
  - Manages non-code resources: strings, graphics, and layout files
- Location manager
  - Access to system location services
- Notification Manager
  - Notify users when events occur
Applications

- Standard apps include:
  - Home – main screen
  - Contacts – contacts database
  - Phone – dial phone numbers
  - Browser – view web pages
  - Email reader – Gmail & others
- Your App!
Lab will help you set up your own laptop for Android programming