Introduction

Programming the Android Platform

CS 282
Principles of Operating Systems II
Systems Programming for Android

Course Goals

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

Administrivia

Logistics

- Douglas C. Schmidt
  - d.schmidt@vanderbilt.edu
  - Office: FGH #226
  - Office hours: M. 1-3pm & W. 1-3pm
    - Nearly always reachable by email

- TAs/graders
  - Zach McCormick
    - zach.mccormick@vanderbilt.edu
    - Office hours: Weekday mornings
  - Jesse Badash
    - jesse.l.badash@vanderbilt.edu
    - Office hours: TBD

- Course URL: www.dre.vanderbilt.edu/~schmidt/cs282

Course Work

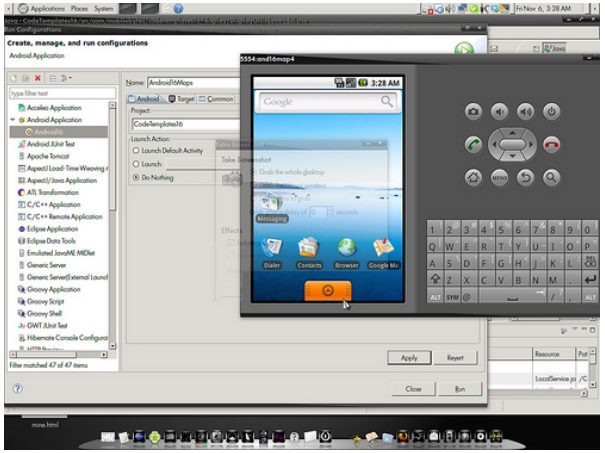
- There will be 5-6 programming assignments written in Java
  - Can use Windows, Linux, Mac, etc.
- ***Must*** be done individually
- Programs will be graded as follows:
  - 40% execution correctness
  - 30% structure (e.g., modularization, information hiding, etc.)
  - 10% insightful programming (e.g., developing reusable class components, etc.)
  - 10% Consistent style (e.g., capitalization, indenting, etc.)
  - 10% appropriate commenting style
- There will be a 5 point deduction (out of a possible 100 points) for each day that your program is late
  - Programs turned in later than two calendar days after the due date will receive a zero
- There will be weekly quizzes & a comprehensive final exam
- The relative weighting of each portion of the course is :
  - 40% Programming projects
  - 40% Quizzes
  - 10% Final Exam
  - 10% Class participation

Ground Rules

- Assignments must be submitted on time
- Work *must* be your own (as per www.owen.vanderbilt.edu/vanderbilt/about-us/honor-code.cfm)
- *No* laptops open in class unless explicitly allowed
- You will be called upon periodically to answer questions
  - 10% class participation grade, so be involved & attend class
- You'll get out of this course what you put into it, so be prepared to work hard & learn a lot
- Be prepared for weekly quizzes & occasional guest lectures
- Make sure to avail yourself of available help, e.g., office hours, TAs, mailing list, etc.

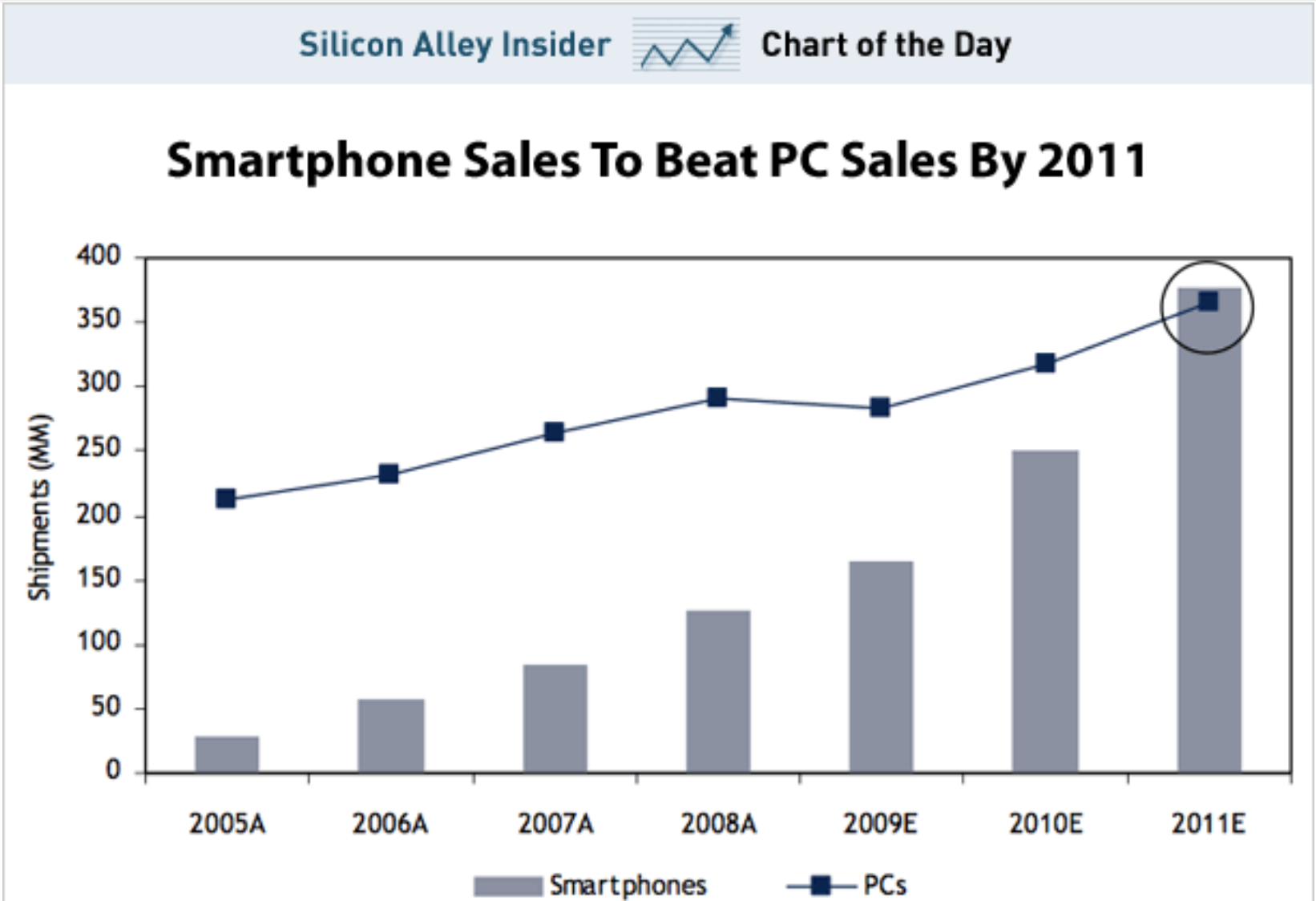
Class Organization

- Mix of lecture & programming exercises
  - ½ presentation
  - ½ laboratory exercises & semester project
- Organization will remain flexible
  - Will change as needed



Why Mobile Devices & Android?

Mobile Devices are the Next Computing Platform



Why Android?

- Android has 50% of the smartphone market (#1)
- iPhone has 30% of the smartphone market (#2)
- Blackberry, Windows Mobile, & Symbian are rapidly losing market share since their platforms not nearly as interesting to develop for as Android/iPhone

Top Smartphone Platforms 3 Month Avg. Ending Feb. 2012 vs. 3 Month Avg. Ending Nov. 2011 Total U.S. Smartphone Subscribers Ages 13+ Source: comScore MobiLens			
	Share (%) of Smartphone Subscribers		
	Nov-11	Feb-12	Point Change
Total Smartphone Subscribers	*100.0%*	*100.0%*	*N/A*
Google	46.9%	50.1%	3.2
Apple	28.7%	30.2%	1.5
RIM	16.6%	13.4%	-3.2
Microsoft	5.2%	3.9%	-1.3
Symbian	1.5%	1.5%	0.0

Why Android?

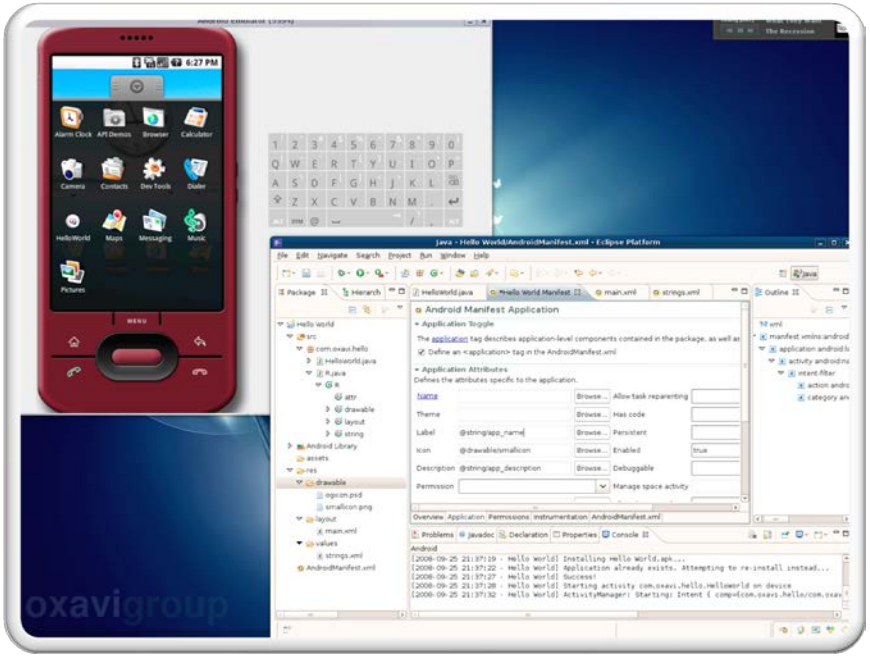
Android is:

- the fastest growing smartphone platform
  - open-source & works on multiple platforms
  - no need to own a Mac
  - no need to join a developer program
- easy to learn for Java (& C++) programmers
  - much easier to transition to than Objective-C

Getting Started with Android

Developing Android Apps

- Android is a software stack for mobile devices that provides an operating system, middleware, & key services/applications
- The Android SDK contains libraries & development tools for creating applications
- Android uses the Eclipse Integrated Development Environment
- Android Eclipse Plugins provide:
  - wizards for creating new apps
  - a visual editor for creating GUIs
  - editors for manipulating Android XML descriptors needed for your app
  - an emulator for testing your apps on your PC
  - a debugger for finding errors in the emulator or on a device



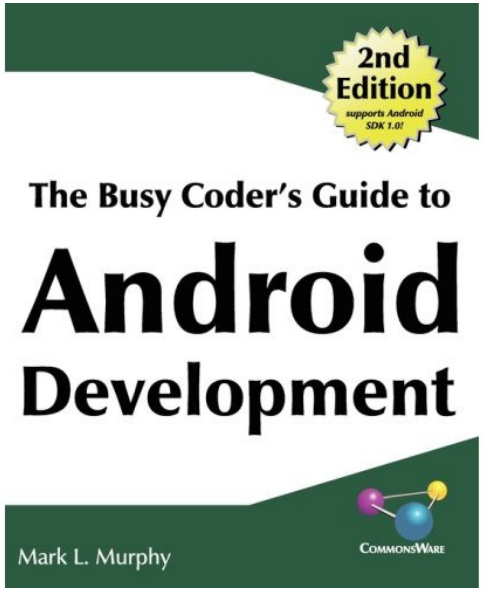
Setting Up an Android Development Environment

- You need to download & install "Eclipse Classic" from: http://www.eclipse.org/downloads/
- You will also need to download & install the Java SDK from http://www.oracle.com/technetwork/java/javase/downloads & the Android SDK from: http://developer.android.com/sdk

- Once Eclipse & the Java & Android SDKs are installed, follow the "Installing the ADT Plugin for Eclipse" instructions here: http://developer.android.com/sdk/installing.html

Figuring Out Android

- Android is well documented
- The Android javadoc references will be critical reference material for your projects:
  - http://developer.android.com/reference/packages.html
- The Android developer guide is another important resource:
  - http://developer.android.com/guide/components
- We recommend "The Busy Coder's Guide to Android Development" e-book
  - http://commonsware.com/warescription

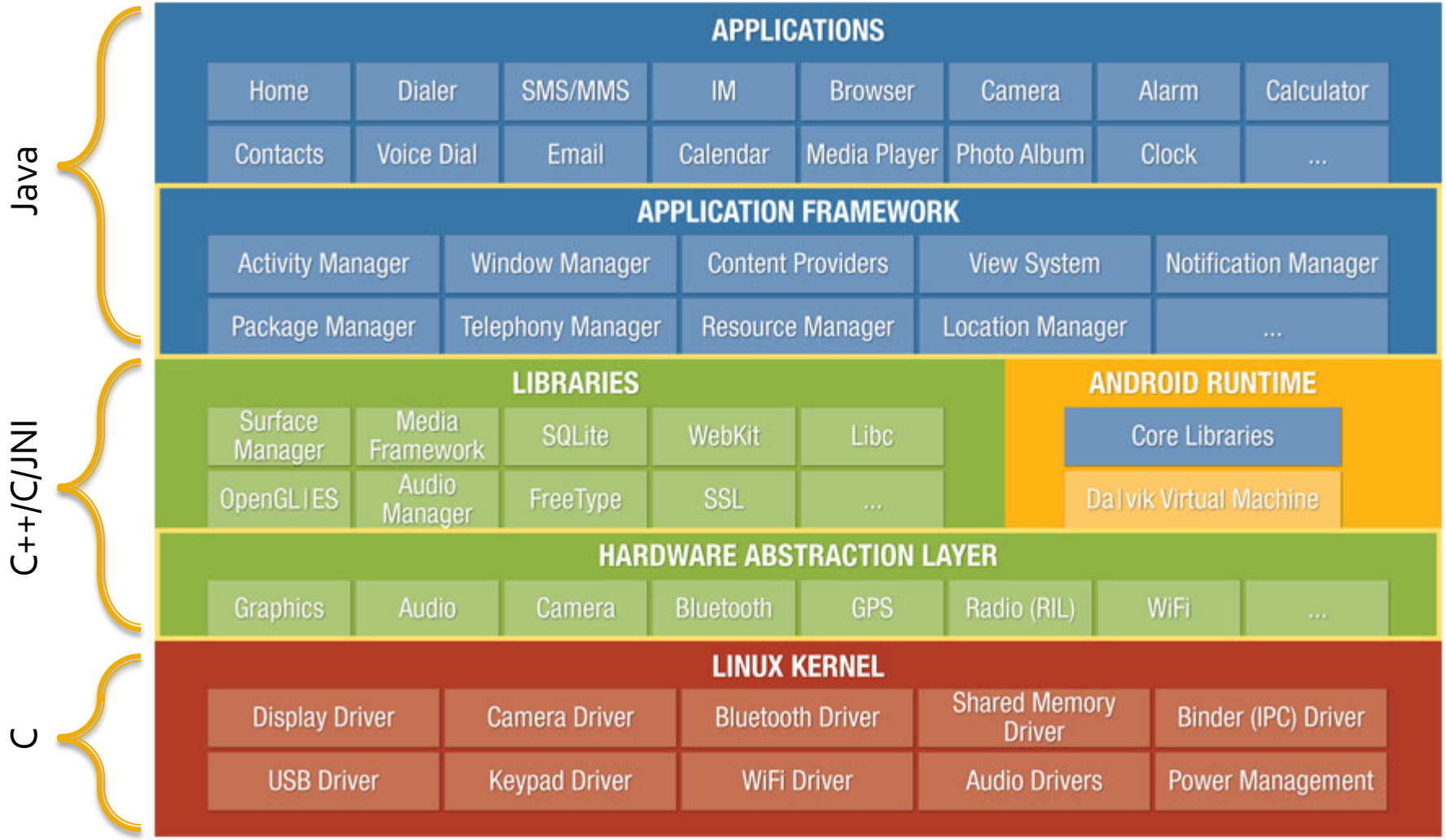


Overview of Android

What is Android?

- Android is a complete software stack for mobile devices (& more)
- Android includes:
  - Operation System
    - Linux variant
  - Specialized Java Virtual Machine
    - Dalvik, which is optimized for power consumption
  - Middleware Stack for:
    - Telephony
    - GUIs
    - Apps
    - App Distribution
    - Etc.

The Android Architecture



http://developer.android.com/guide/basics/what-is-android.html

Linux Kernel

- Abstraction layer between hardware & software
- Provides services such as:
  - Security
  - Memory & process management
  - Network stack
  - Device driver model

Linux Kernel (cont.)

- Android-specific components
  - Binder – inter-process communication (IPC)
  - Android shared memory
  - Power management
  - Alarm driver
  - Low memory killer
  - Kernel debugger & Logger

Hardware Abstraction Layer

- User space C/C++ library layer
- Defines the interface that Android requires hardware "drivers" to implement
- Separates Android platform logic from hardware interface

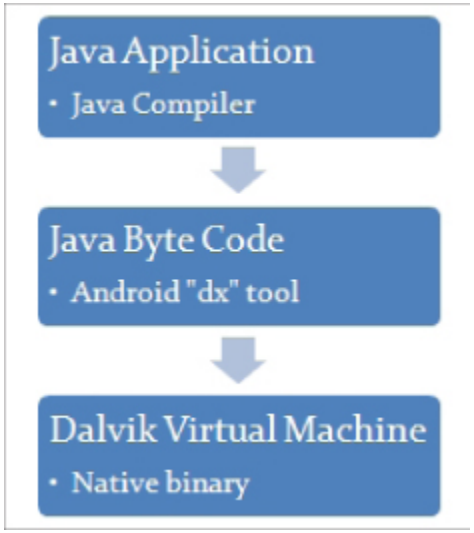
- Why a user-space HAL?
  - Not all components have standardized kernel driver interfaces
  - Kernel drivers are GPL, which exposes any proprietary IP
  - Android has specific requirements for hardware drivers

Native C/C++ Libraries

- System C library
  - bionic libc
- Surface Manager
  - display management
- Media Framework
  - audio/video
- FreeType
  - library for rendering fonts
- 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 (Java) Libraries
  - Dalvik Virtual Machine



Core (Java) Libraries

- Core Java classes
  - android.*
  - java.*, javax.*
  - junit.*
  - org.apache.*, org.json.*, org.xml.*
- Doesn't include all standard Java SDK classes
  - http://developer.android.com/reference/packages.html
  - http://www.zdnet.com/blog/burnette/java-vs-android-apis/504

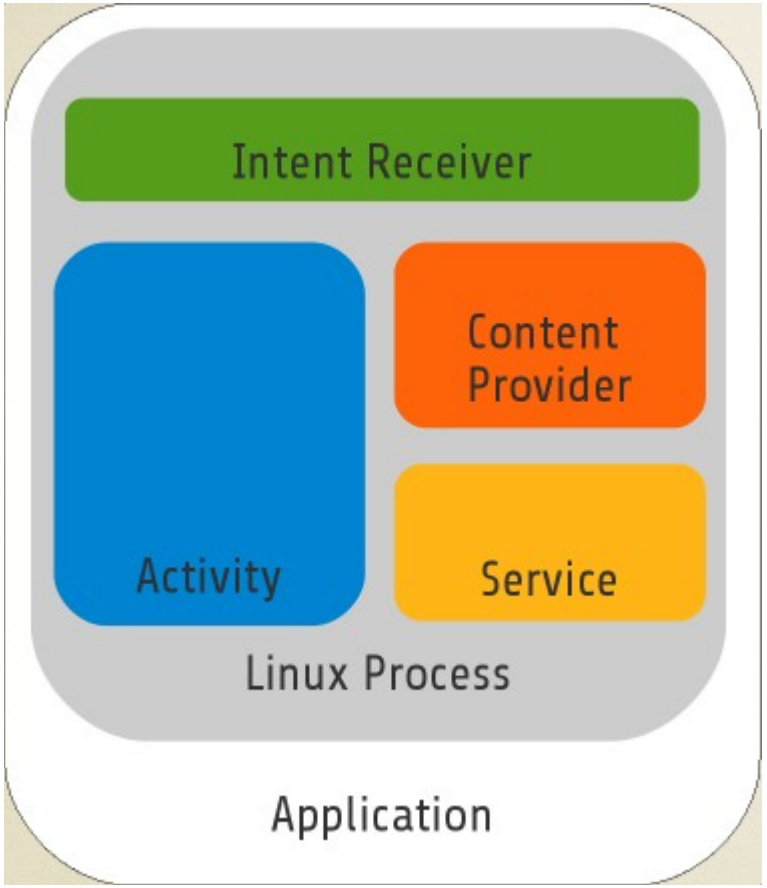
Dalvik Virtual Machine

- Android apps 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

http://sites.google.com/site/io/dalvik-vm-internals

Key Android App Components

- Activities
  - represents a single screen with a user interface
- Services
  - runs in the background to perform long-running operations or to perform work for remote processes
- Content Providers
  - manages a shared set of application data
- Broadcast Receivers
  - a component that responds to system-wide broadcast announcements



Application Frameworks (cont.)

- 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 & common navigation stack

Application Frameworks (cont.)

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

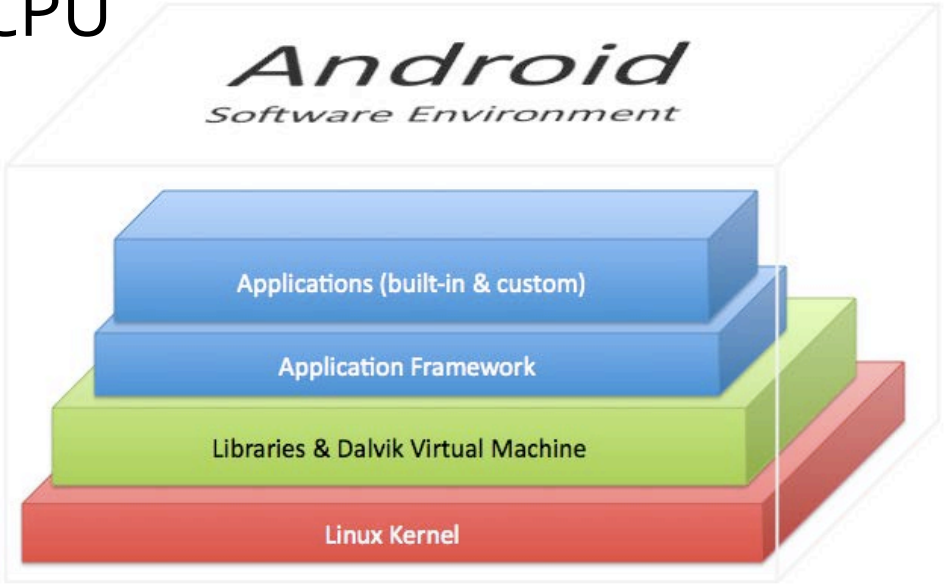
Applications

- Standard apps include:
  - Home – main screen
  - Contacts – contacts database
  - Phone – dial phone numbers
  - Browser – view web pages
  - Email reader – Gmail & others
- Your App!

Innards of the Dalvik VM

Dalvik Virtual Machine

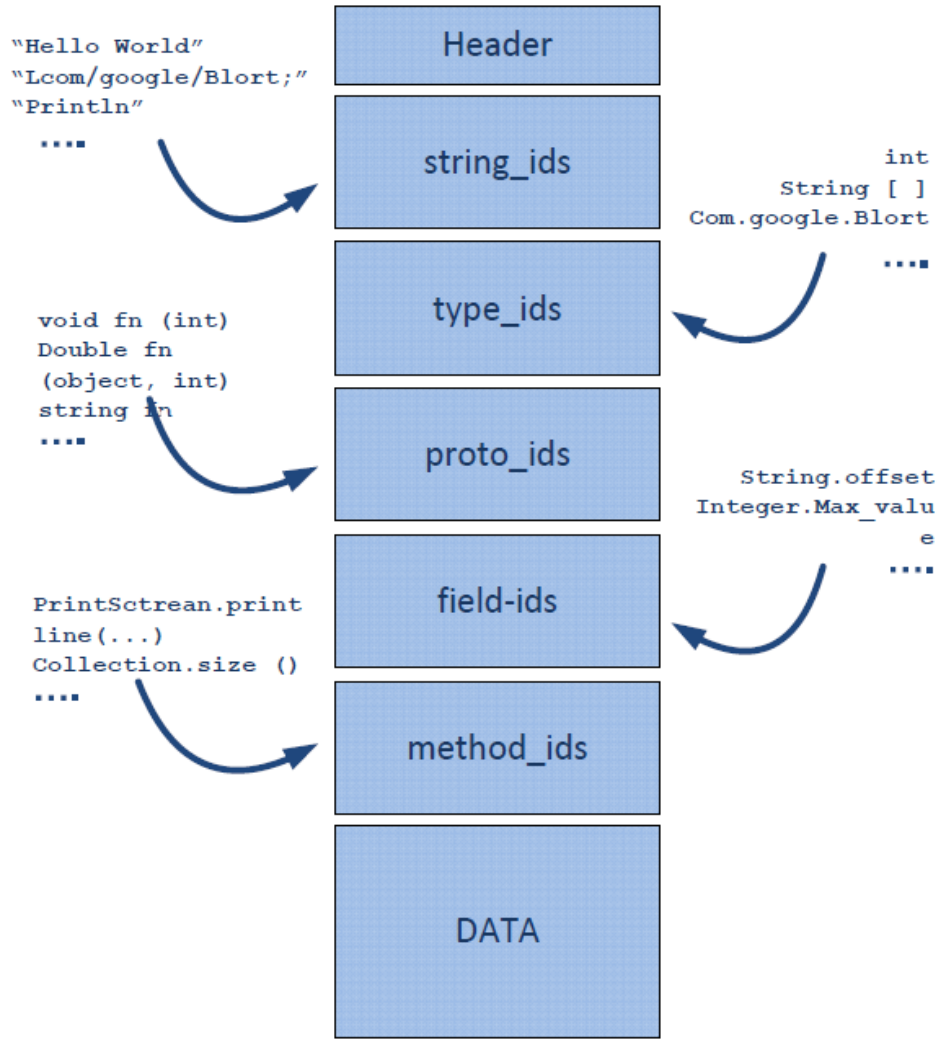
- Dalvik VM designed explicitly to run on a handset
  - Originally relatively little RAM
    - e.g., 64Mb total: ~40Mb for Linux & Android services, ~10Mb for Android middleware, ~10Mb available at runtime for apps
  - Originally relatively slow CPU
  - No swap space
  - Limited battery life
  - Multiple independent, mutually-suspicious processes



http://sites.google.com/site/io/dalvik-vm-internals

Dalvik Virtual Machine (cont.)

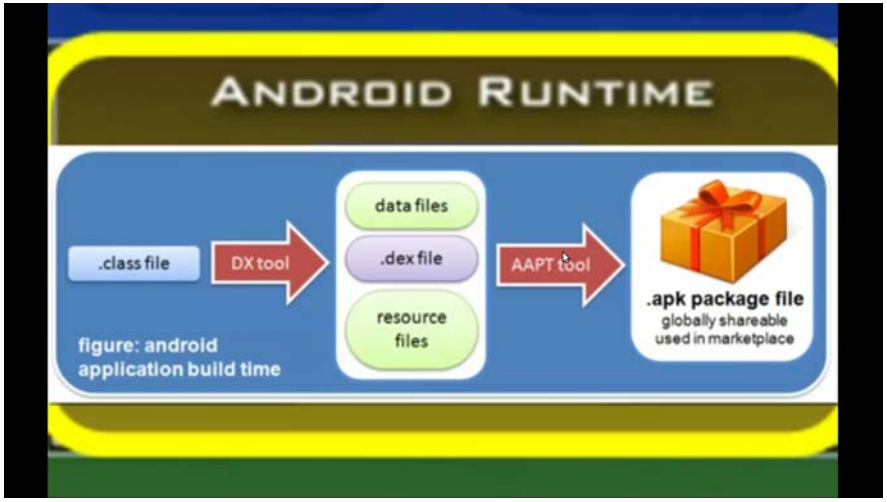
- Memory
  - .dex file has common constant pool for multiple classes
  - Modified garbage collection to improve memory sharing
- CPU
  - Optimizations at installation time
  - Register-based, rather than stack-based

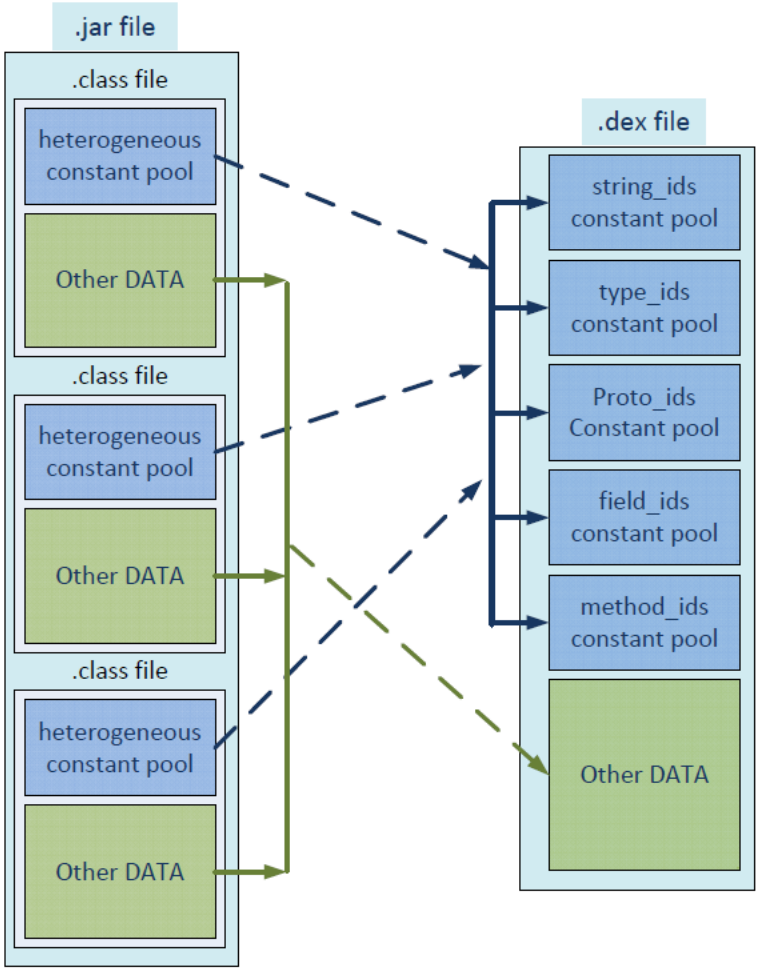


http://imsciences.edu.pk/serg/wp-content/uploads/2010/10/1st_Analysis-of-Dalvik-VM.pdf

Converting Java byte-code to Dalvik Byte-code

- Every Java program is compiled to byte-code
- The Java byte-code is then transformed into Dalvik byte-code with the help of the dx tool & stored in .dex file
- That's upon what Dalvik performs operations, such as verification & optimization





http://imsciences.edu.pk/serg/wp-content/uploads/2010/10/1st_Analysis-of-Dalvik-VM.pdf

Why Dalvik Uses Registers

- Expected benefits over stack-based VMs
  - Avoids slow instruction dispatch
  - Avoids unnecessary memory accesses
  - More efficient instruction stream
    - Higher semantic density per instructions
- 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 http://www.youtube.com/watch?v=ptjedOZEXPM

Dalvik VM Example

```
public static long sumArray(int[] arr) {
    long sum = 0;
    for (int i : arr) {
        sum +=    i;
    }
    return sum;
}
```

Java Bytecode

```
0000: lconst_0
0001: lstore_1
0002: aload_0
0003: astore_3
0004: aload_3
0005: arraylength
0006: istore 04
0008: iconst_0
0009: istore 05
000b: iload 05              // rl ws
000d: iload 04              // rl ws
000f: if_icmpge 0024        // rs rs
0012: aload_3               // rl ws
0013: iload 05              // rl ws
0015: iaload                // rs rs ws
0016: istore 06             // rs wl
0018: lload_1               // rl rl ws ws
0019: iload 06              // rl ws
001b: i2l                   // rs ws ws
001c: ladd                  // rs rs rs rs ws ws
001d: lstore_1              // rs rs wl wl
001e: iinc 05, #+01         // rl wl
0021: goto 000b
0024: lload_1
0025: lreturn
```

read local

write local

read stack

write stack

- 25 bytes
- 14 dispatches
- 45 reads
- 16 writes

% javap –c ClassName

Dex Bytecode

% dexdump –d classes.dex

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

- 18 bytes
- 6 dispatches
- 19 reads
- 6 writes

Loop Wisely to Conserve Power

1. `for (int i = initializer; i >= 0; i--)`
2. `int limit = calculate limit;`
   `for (int i = 0; i < limit; i++)`
3. `Type[] array = get array;`
   `for (Type obj : array)`
4. `for (int i = 0; i < array.length; i++)`
5. `for (int i = 0; i < this.var; i++)`
6. `for (int i = 0; i < obj.size(); i++)`
7. `Iterable<Type> list = get list;`
   `for (Type obj : list)`

Assignment

- Lab 1 will help you set up your own laptop for Android programming