L#2. Mobile Application Introduction

Introduction

Requirement

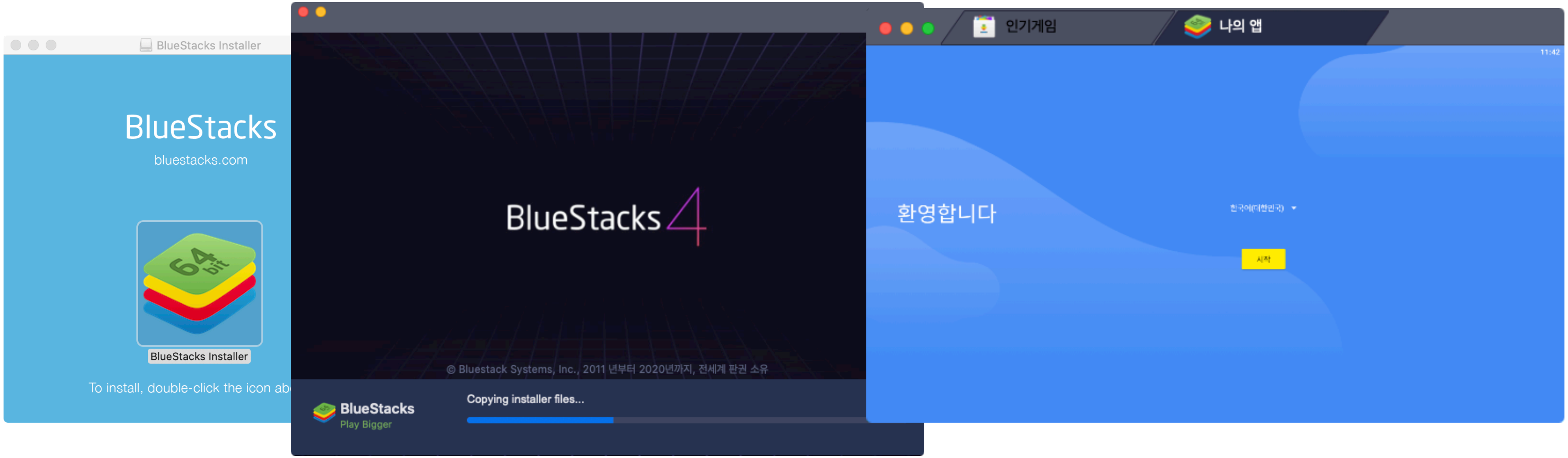
- For this class
  - You need an Android-base smartphone
    - Actually you can use AVD or some 3rd party Android base VM
    - But it’s relatively or very slow
    - The best choice is Non-rooted and Rooted device

Requirement

- For this class
  - You need an Android-base smartphone
    - Without physical devices → Genymotion, Bluestacks, Nox, Koplayer, AndY, MeMU
    - Downside is for Google Play → But there is a solution

Requirement

- BlueStacks



To install, double-click the icon ab

© Bluestack Systems, Inc., 2011 년부터 2020년까지, 전세계 판권 소유

Copying installer files...

환영합니다

Smartphone

- Omnia(2008)
- iPhone(2007)
- GS S(2010)

Smartphone

- Many chance for developers
  - New platforms and new language → New World

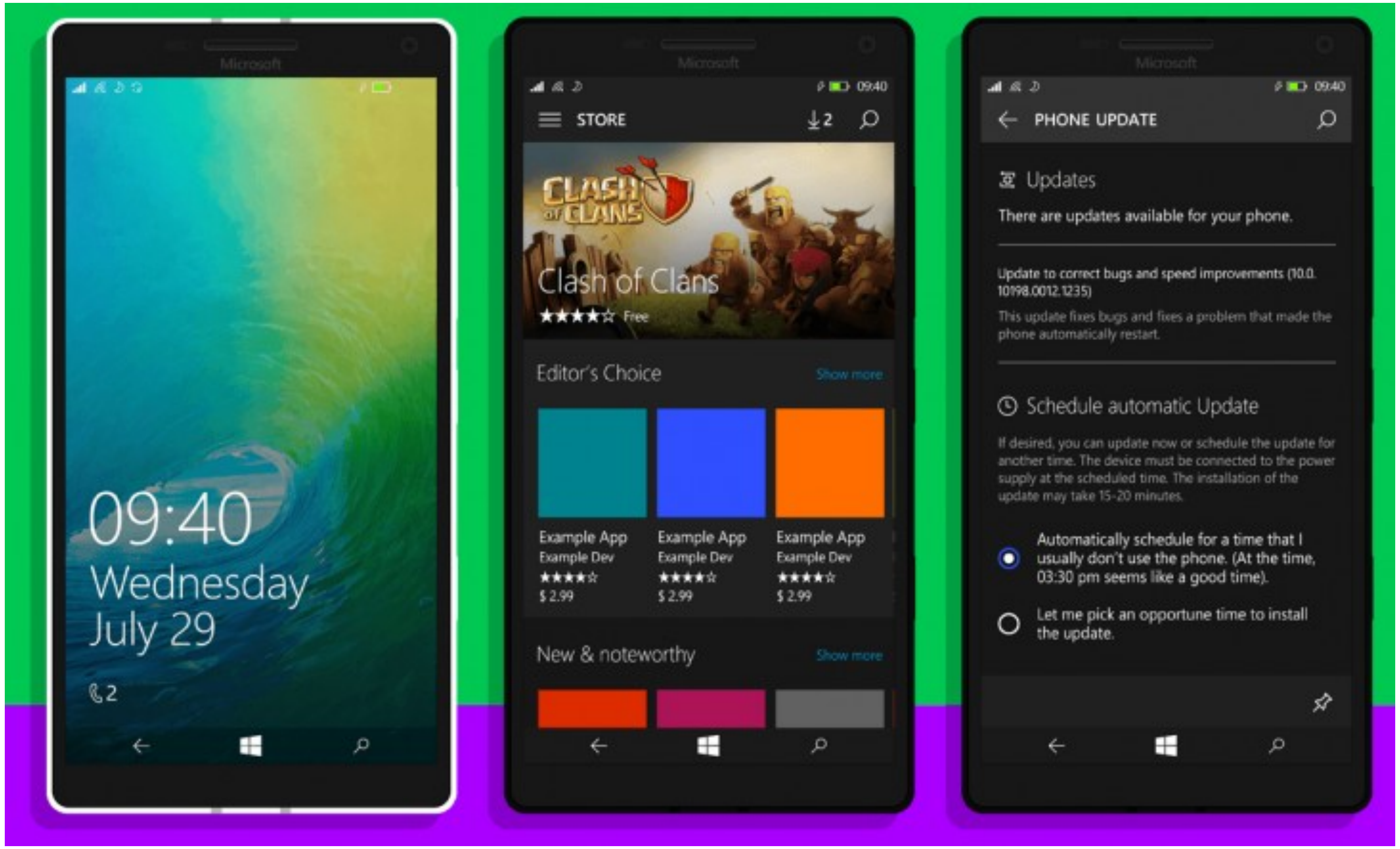
Smartphone

- Platform
    - iOS vs. Android(AOS)

    - iOS(Apple)
        - Closed environment(Only Apple)
            - Apple Store, Closed System
        - Jailbreak iOS
    - AOS(Google)
        - Open-Source
        - Many vendor(!Google)

Smartphone

- Platform : Microsoft



Updates

There are updates available for your phone.

Update to correct bugs and speed improvements (10.0. 10198.0012.1235)

This update fixes bugs and fixes a problem that made the phone automatically restart.

Schedule automatic Update

If desired, you can update now or schedule the update for another time. The device must be connected to the power supply at the scheduled time. The installation of the update may take 15-20 minutes.

● Automatically schedule for a time that I usually don't use the phone. (At the time, 03:30 pm seems like a good time).

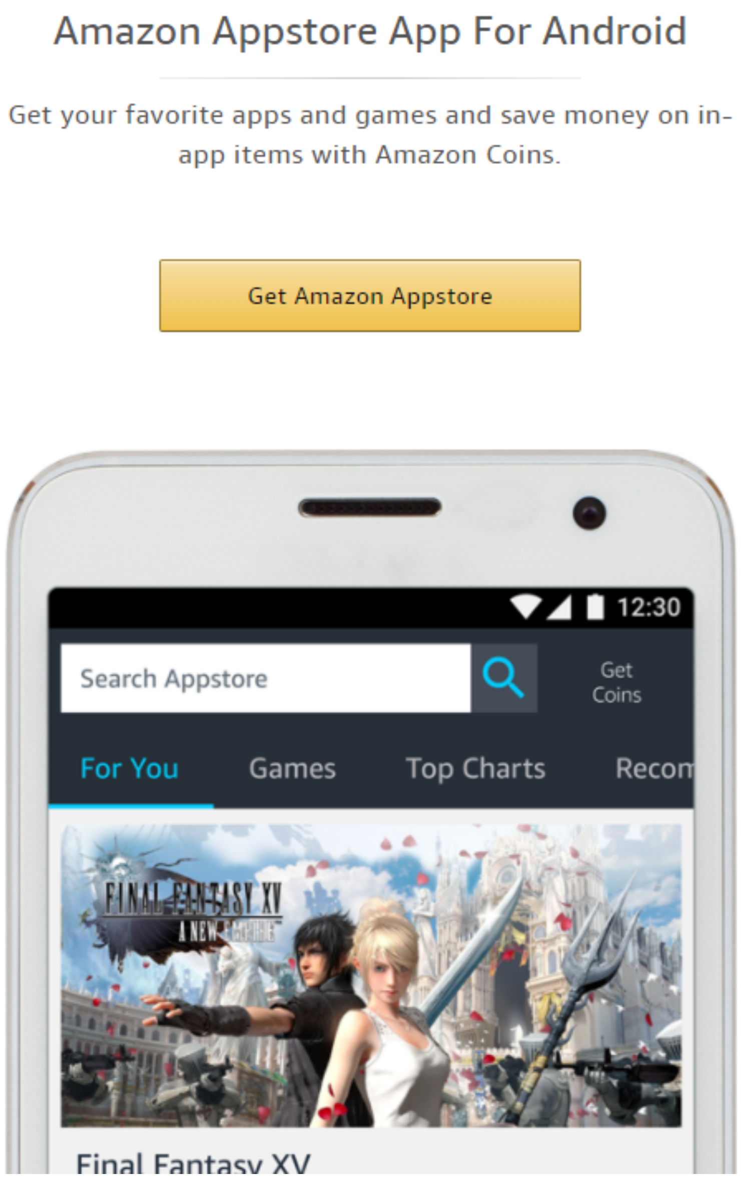
○ Let me pick an opportune time to install the update.

Smartphone

- Platform
    - Android / Java, C, C++ / Android Studio(Eclipse) / Open Source
    - iOS / Swift, Objective-C / X Code / Apple
    - Windows Mobile / C# / Visual Studio
    - BlackBerry / Java, C, C++

Smartphone

- App Distribution
  - iOS : Only Apple App Store
  - AOS
    - Google Play, Amazon App
    - 3rd Party Store(China)



Amazon Appstore App For Android

Get your favorite apps and games and save money on in-app items with Amazon Coins.

Android History

- History
  - 2008 1.0 Beta; 1.5 Cupcake; 1.6 Donut, Sep 2009
  - 2009 2.0/2.1 (Éclair); revamped UI, introduced HTML5, W3C Geolocation API and Exchange ActiveSync 2.5 support
  - 2010 2.2 (Froyo), May 2010; speed improvements with JIT optimization and the Chrome V8 JavaScript engine, added Wi-Fi tethering
  - 2010 2.3 (Gingerbread), Dec 2010; refined the UI, improved keyboard and copy/paste features, Near Field Communication
  - 2011 3.0/3.1 (Honeycomb), Feb 2011; a tablet-oriented release

Android History

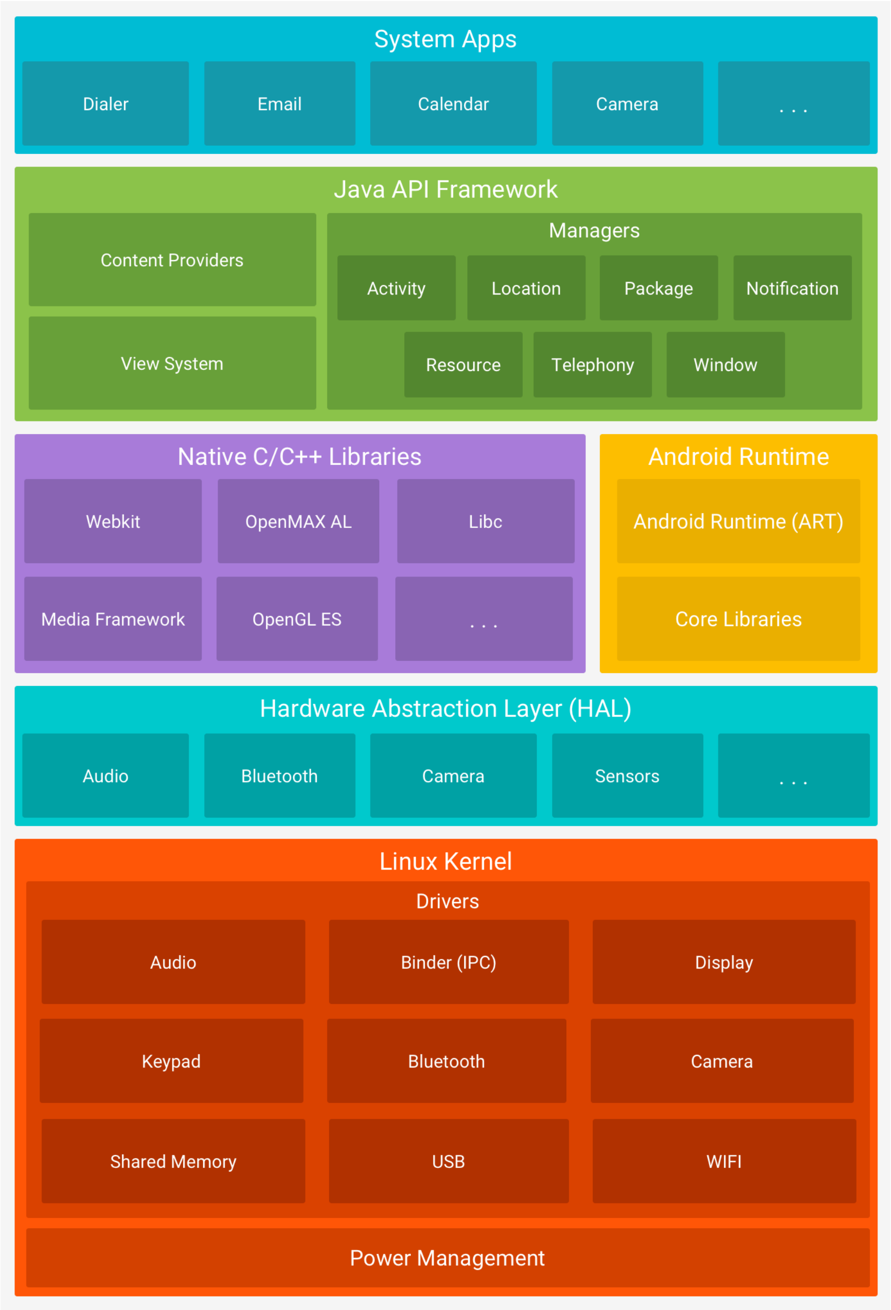
- History
  - 2011 4.0 (Ice Cream Sandwich)
  - 2012 4.1~4.3 Jelly Bean (API level 16~18)
  - 2013~2014 4.4 KitKat (API level 19~20)
  - 2014~2015 5.0~5.1 Lollipop (API level 21~22)
  - 2015 6.0 Marshmallow (API level 23)
  - 2016 7.0~7.1 Nougat (API level 24~25)
  - 2017 8.0 Oreo (API level 26~27)
  - 2018 9.0 Pie (API level 28)
  - 2019 10 Q(API level 29)
  - 2020 11

Android System

- Open source software for mobile or wearable devices
- Complete stack : Operating System, Middleware, Application
- Powered by Linux Operating System
- Almost application development in Java(SDK)

Architecture

- Android Platform Architecture
  - Linux Kernel
  - Native Library
  - Android Runtime
  - Java API Framework

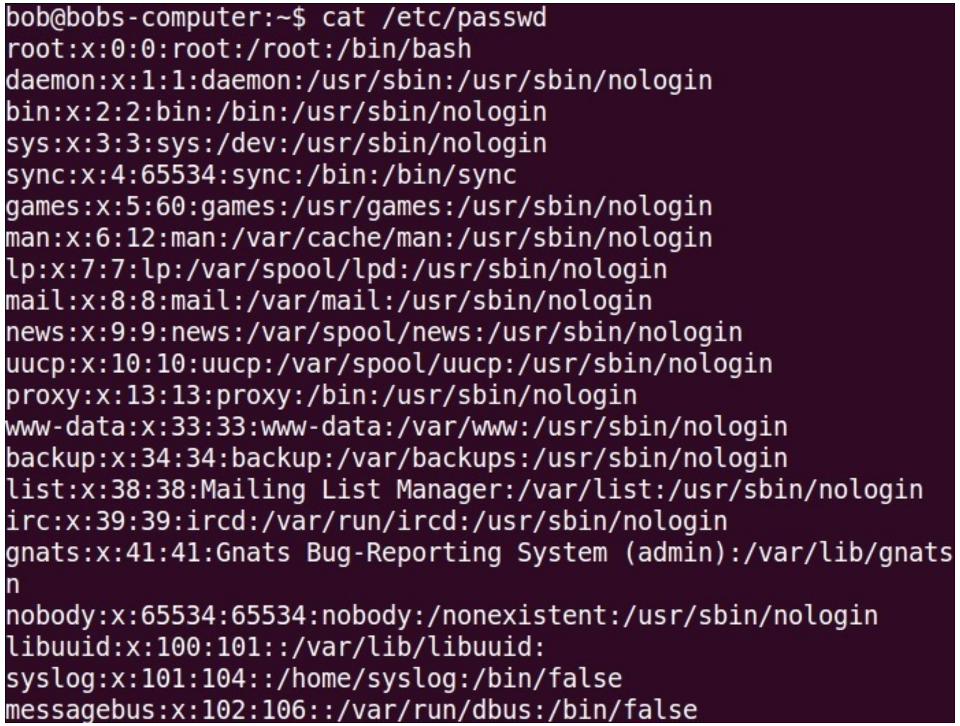


Android vs. Java

- Syntax is the same : Android == Java
- Android API include almost Java API → Some lib is not included.
- Android SDK = Java SE – AWT/Swing + Android API

Android AppId

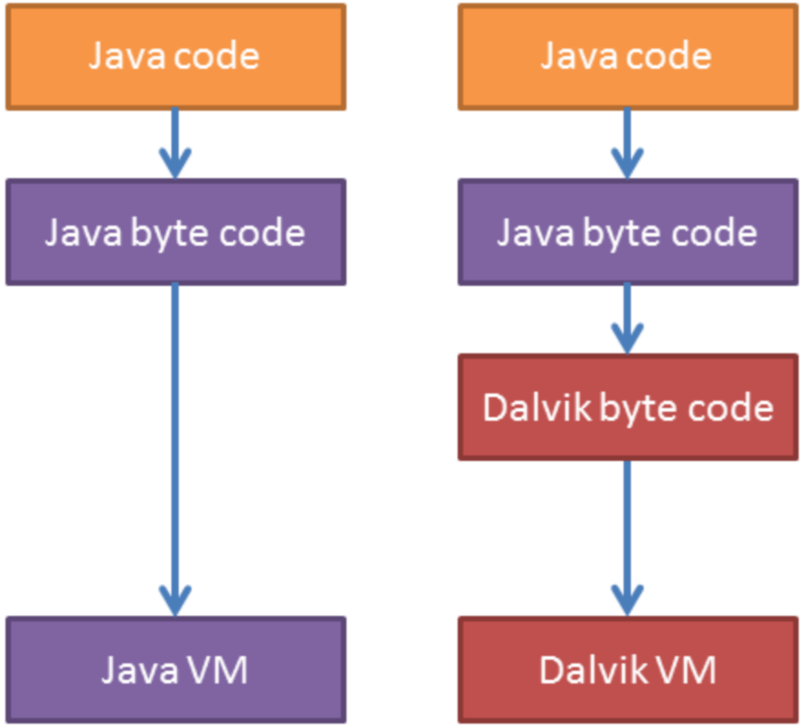
- Have its own AppId like Linux user ID
  - Based on AppId → Like sandbox, Access control, Isolation



```
bob@bobs-computer:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats
n
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:106::/var/run/dbus:/bin/false
```

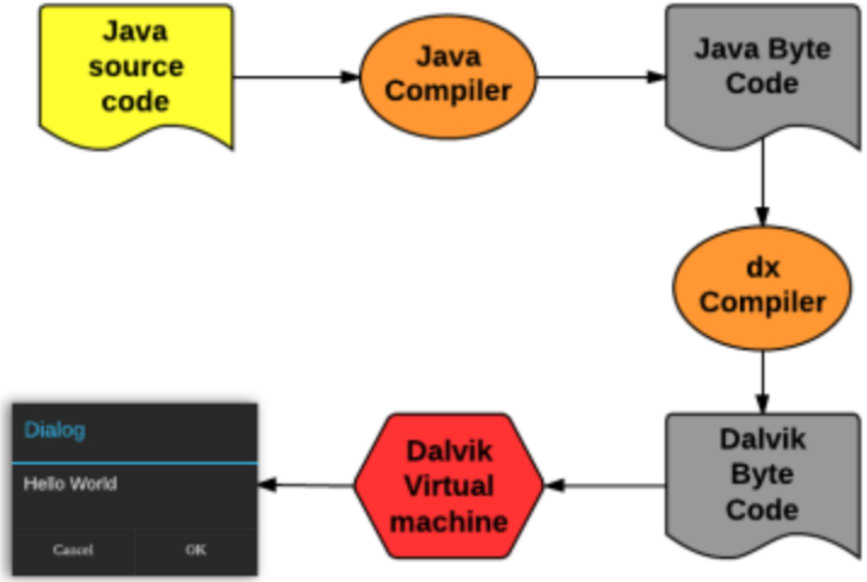
Android Runtime

- Android based on Android API which almost like JAVA
- Java application over JavaVM → Android application over Dalvik



Android Runtime

- Dalvik VM is a new JVM by Google
  - Register-based versus stack-based JVM
  - Different set of Java libraries than JDK
- Dalvik VM has been optimized for mobile devices
  - not so powerful CPU
  - memory shortage
  - Dalvik Executable .dex format is compact
  - run multiple VMs efficiently.



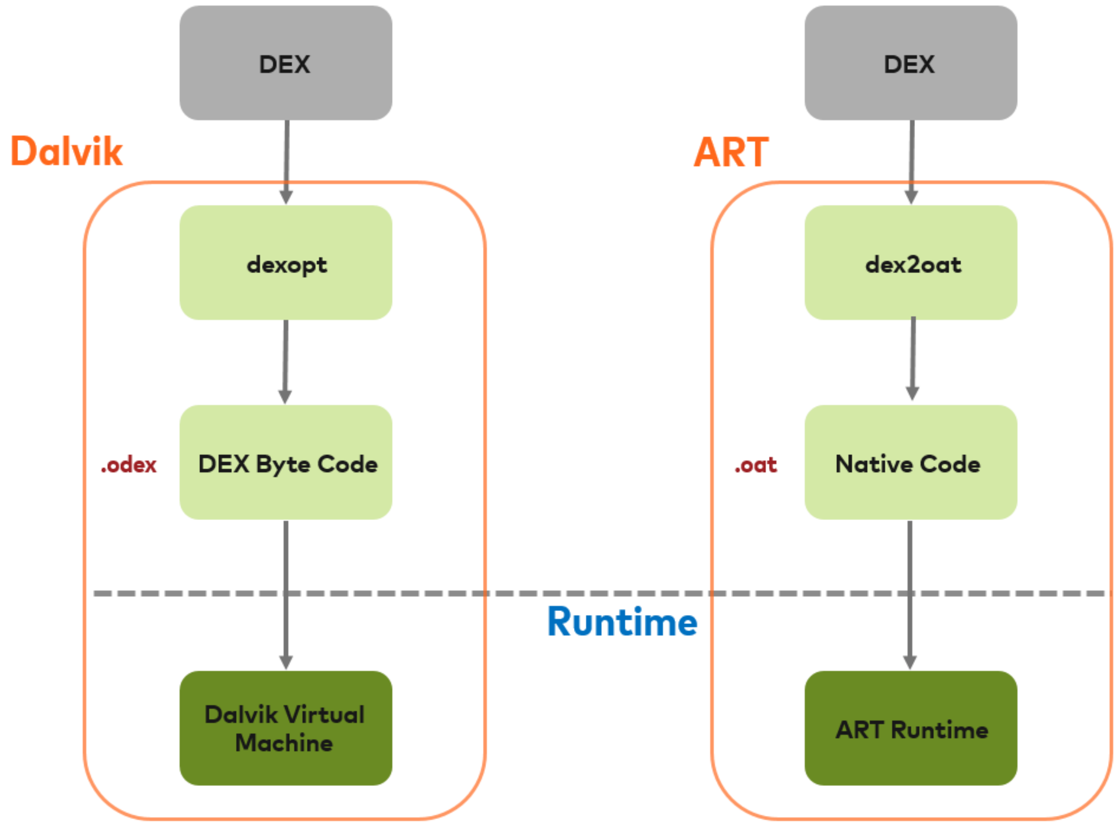
Android Runtime

- Dalvik vs. ART
    - Dalvik : < Android 5.0
    - ART : >= Android 5.0

    - ART and Dalvik are compatible runtimes running Dalvik bytecode
    - Apps developed for Dalvik should work when running with ART
        - some techniques that work on Dalvik do not work on ART.
        - ART→Dalvik, !Dalvik→ART

Android Runtime

- Dalvik vs. ART



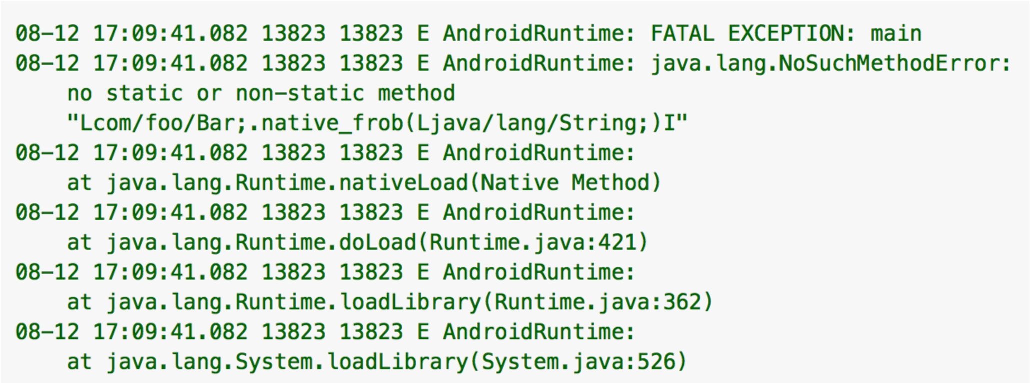
Dalvik vs ART

Android Runtime

- Dalvik vs. ART
  - ART→Dalvik, !Dalvik→ART

  - GC(Garbage Collector) on Dalvik with System.gc() is unnecessary
    - System.getProperty("java.vm.version") >= "2.0.0"

  - Exception handling
    - NoSuchMethodError, NoSuchFieldError



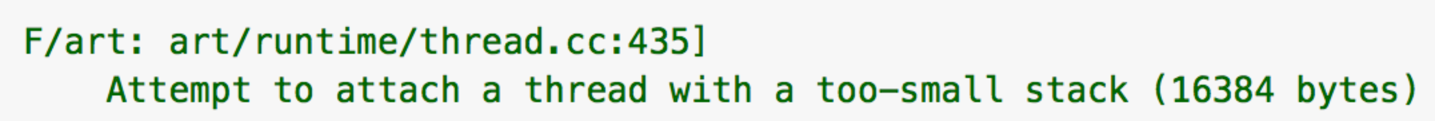
```
08-12 17:09:41.082 13823 13823 E AndroidRuntime: FATAL EXCEPTION: main
08-12 17:09:41.082 13823 13823 E AndroidRuntime: java.lang.NoSuchMethodError:
    no static or non-static method
    "Lcom/foo/Bar;.native_frob(Ljava/lang/String;)I"
08-12 17:09:41.082 13823 13823 E AndroidRuntime:
    at java.lang.Runtime.nativeLoad(Native Method)
08-12 17:09:41.082 13823 13823 E AndroidRuntime:
    at java.lang.Runtime.doLoad(Runtime.java:421)
08-12 17:09:41.082 13823 13823 E AndroidRuntime:
    at java.lang.Runtime.loadLibrary(Runtime.java:362)
08-12 17:09:41.082 13823 13823 E AndroidRuntime:
    at java.lang.System.loadLibrary(System.java:526)
```

Android Runtime

- Dalvik vs. ART
  - ART→Dalvik, !Dalvik→ART

  - Stack size
    - Separate stack on Dalvik : Java/Native
    - Java stack size = 32kb, Native stack size = 1mb

    - One stack on ART
      - StackOverflow : Touch a memory of VM(VirtualMachineError)
      - Too-small stack by AttachCurrentThread()
        - pthread_attr_setstack(), pthread_attr_setstacksize()



```
F/art: art/runtime/thread.cc:435]
    Attempt to attach a thread with a too-small stack (16384 bytes)
```

Android Runtime

- Dalvik vs. ART
    - ART→Dalvik, !Dalvik→ART

    - Strict bytecode exmination by ART
        - Some techniques onDalvik isn’t allowed on ART
        - With AndroidStudio, there is not Error. But after compilation with APK, pro-handling will make an Error(Ex. Obfuscation)

        - invalid control flow
        - unbalanced monitorenter/moniterexit
        - 0-length parameter type list size

Android Runtime

- Dalvik vs. ART
    - ART Features

    - Ahead-of-time(AOT) Compilation
    - Improved GC
    - Development and debugging improvements
    - Support for sampling profiler
    - Support for more debugging features
    - Improved diagnostic detail in exceptions and crash reports

Android Runtime

- Dalvik vs. ART
    - ART Features : Ahead-of-time(AOT) Compilation

    - Will be able to improve app runtime performance
    - Tighter verification

    - AOT vs JIT(Just-In-Time)

Android Runtime

- Dalvik vs. ART
  - ART Features : Ahead-of-time(AOT) Compilation

  - When an app is launched, Dalvik bytecode is translated into machine code with JIT
  - On the ART, when an app is installed on the device, the bytecode is translated into machine code with ART
    - DEX2OAT

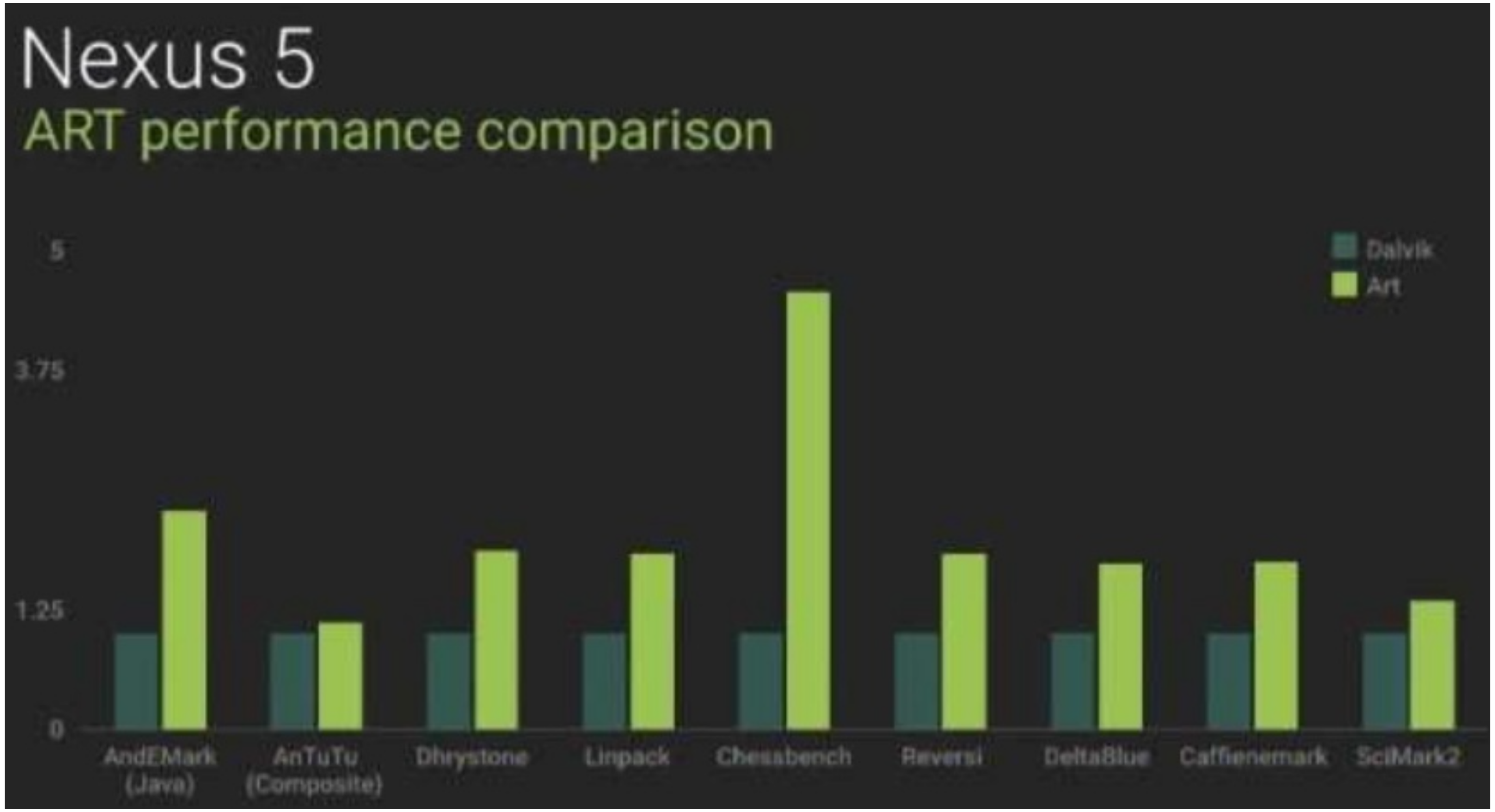
Android Runtime

- Dalvik vs. ART
  - ART Features : Ahead-of-time(AOT) Compilation

  - Comparison
    - Dalvik
      - Lower stoarge, Space cossumption from JIT
      - Take a time for cache, Fater booting time
      - Lower internal storage(Old model)
      - Stable…
    - ART
      - Fast load time, lower processor usage, short installation time
      - Long booting time
      - More internal storage → Compiled app + APK
      - Unstable???

Android Runtime

- Dalvik vs. ART
  - ART Features : Ahead-of-time(AOT) Compilation



Android Runtime

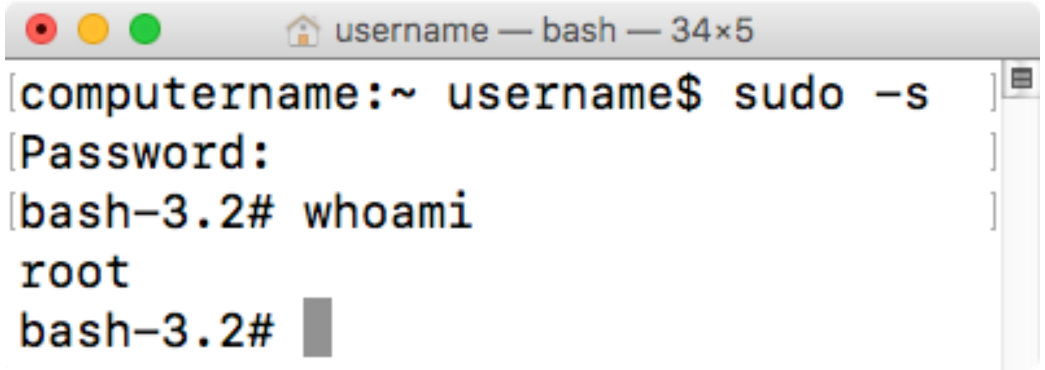
- Dalvik vs. ART
  - ART Features : Ahead-of-time(AOT) Compilation
  - Free processor from DEX translation to machine code during the app's execution → Low battery consumption
  - On the ART, longer battery recharge interval

Android Rooting

- Android devices such as tabs, Smartphone's etc., by default comes with a set of restricted set of permissions for its users
- By rooting an android device a user can bypass all the restrictions implemented by hardware manufactures and carriers

Android Rooting

- The term “root” comes from the Unix/Linux world and is used to describe a user who has “super user” rights or permissions to all the files and programs in the OS



```
computername:~ username$ sudo -s
Password:
bash-3.2# whoami
root
bash-3.2#
```

Android Rooting

- Advantage of Rooting
  - Increasing Battery Life
  - Using Custom Recoveries
  - Using Custom Rom's
  - Increasing speed of the device

Android Rooting

- Disadvantage of Rooting
  - Bricking or breaking your device beyond repair
  - Security (Malware, Virus)
  - Voiding Warranty (in some cases)
  - Lose of data

Android Rooting

- How to Rooting
  - The Process of Rooting an android device varies based on the model of your device
  - By either googling or searching XDA forums
    - https://www.xda-developers.com/root/

Android Rooting

- Before Rooting
  - Perform full backup before rooting
  - You may void warrany
  - You may brick your device
  - You do it on your own responsibility

WARNING

Q&A