CS260 Intro to Java & Android
04. Android Intro

Winter 2015
Android - Getting Started

- Android SDK contains:
  - API Libraries
  - Developer Tools
  - Documentation
  - Sample Code

- Present development tools:
  - Eclipse with the Android Developer Tool (ADT) plugin which integrates developer tools
  - Android Studio
Android Portability

- Android applications run within the Dalvik virtual machine
- ART is a new Android runtime being introduced in 4.4

Development Platforms:
- Windows (XP, Windows, 7, 8)
- Linux
- Mac OS 10.4.8 or later (Intel chips only)
Android HelloWorld Application

- Start Android Studio
- We will create our warm fuzzy HelloWorld
New Android Project
Click “Next” takes us to
Click “Next” takes us to
Click “Finish” takes us to
Run the Android Application

- Special keys
  - left ctrl & F11 - landscape
  - Esc - back button
  - Home - Home
  - F3 - Call / Dial button
  - F4 - Hang up / end call
  - F5 - Search
- More Shortcuts
Design Mode
Hello World Android Project

```java
package helloworld.cs.pacificu.edu.helloworld;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here.
        // The action bar closes automatically if a
        // menu item is clicked.
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```
Creating Virtual Devices
Important Android Dates

- Google acquires Android, August 2005
- Open Handset Alliance (OHA) announced, November 2007. OHA developed Android and is “...committed to commercially deploy handsets and services using the Android Platform.” [10]
- First Android Phone, G1, October 2008
- Android SDK 1.0, October 2008
What is Android?

- Android is a software stack (set of programs working together) for mobile devices that includes:
  - an operating system
  - middleware
  - applications
Android Architecture

![Android Architecture Diagram]

- **Applications**
  - Home
  - Contacts
  - Phone
  - Browser
  - ...

- **Application Framework**
  - Activity Manager
  - Window Manager
  - Content Providers
  - View System
  - Package Manager
  - Telephony Manager
  - Resource Manager
  - Location Manager
  - Notification Manager

- **Libraries**
  - Surface Manager
  - OpenGL | ES
  - SGL
  - Media Framework
  - FreeType
  - SSL
  - SQLite
  - WebKit
  - libc

- **Android Runtime**
  - Core Libraries
  - Dalvik/Virtual Machine

- **Linux Kernel**
  - Display Driver
  - Camera Driver
  - Flash Memory Driver
  - Binder (IPC) Driver
  - Keypad Driver
  - WiFi Driver
  - Audio Drivers
  - Power Management
Linux Kernel

- Android relies on Linux version 2.6 (3.x from Android 4.0 Ice Cream Sandwich) for:
  - memory management
  - process management
  - security
  - networking
- You will not make Linux system calls
- Some utilities interact with Linux
  - e.g. adb shell
With an emulator running, open a Windows command shell

- Type `adb shell`
- Type `ls`

Now you can examine the Linux file system of the phone which aids in debugging.
Native Libraries

- The native libraries are written in C & C++
- The libraries are exposed through the Application framework
Application Framework

- Android developers have access to the same framework APIs use by the core applications.

- Services and systems for applications include:
  - **Views** – including lists, grids, buttons, ....
  - **Content Providers** – methods for accessing data
  - **Resource Manager** – organizes non-code resources such as strings and layout files
  - **Notification Manager** – displays custom alerts
  - **Activity Manager** – manages lifecycle of applications
Android Runtime

Every Application:

- Runs in its own process space
- Has a separate instance of the Dalvik VM
  - The Dalvik VM uses the Linux kernel for functionality such as threading and low-level memory management
  - Dalvik VM != JVM
- All Android code is written in Java and run within the Dalvik VM
What is Dalvik?

- Dalvik is a VM optimized for low memory requirements
- Android code is compiled into bytecodes executed by the Dalvik VM
- bytecodes are machine-independent instructions
Android Applications

- Apps are written in Java
- Code is compiled into Android package (.apk file)
- All code (including data & resource files) in .apk is one application
Android Application Specifics

- Android is a multi-user Linux system where each application is a user
- Only one application is visible at a time
- Each process has its own VM running an application in isolation
- Two or more applications can share data
- Applications consist of one or more activities
What is an Activity?

• An Activity represents a single screen with a UI
• Ex: Email Application consists of activities for
  • Showing list of emails
  • Composing an email
  • Reading an email
• Each activity is independent
• Other applications can use a particular activity if the email application gives permission to do so
Activity Lifecycle

Activity – a process that performs some specific action

- Every Android application is made up of one or more activities managed on an Activity Stack (AS) or the “back stack”.

- A new activity is always placed on top of the AS and then becomes the running activity.

- The AS is LIFO; therefore, when the Back button is pressed the current activity is popped and destroyed
Activity Lifecycle Visual
Activity States

An activity has essentially four states:

- **running** – in the foreground of the screen
- **paused** – lost focus but still visible with all state maintained
  - How? A new activity that is transparent or not full sized is running on top of the stack
- **stopped** – a new activity completely obscures another activity
  - The stopped activity is no longer visible
  - State is maintained
- **destroyed** – the activity must be completely restarted and the state information must be
Activity Skeleton

```java
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // The activity is being created
        super.onCreate(savedInstanceState);
    }

    @Override
    protected void onStart () {
        // The activity is about to become visible
        super.onStart ();
    }

    @Override
    protected void onResume () {
        // The activity has become visible (is is now "resumed")
        super.onResume ();
    }

    @Override
    protected void onPause () {
        // Another activity is taking focus
        super.onPause ();
    }

    @Override
    protected void onStop () {
        // The activity is no longer visible (it is now "stopped")
        super.onStop ();
    }

    @Override
    protected void onDestroy () {
        // The activity is about to be destroyed
        super.onDestroy ();
    }

    @Override
    protected void onRestart () {
        // The user returns to the activity
        super.onRestart ();
    }
}
```
ActivityLifeCycleDemo Application

Copy the Android Project ActivityLifeCycle from CS260-01Public

1. Place the file in AndroidStudioProjects on your local machine
2. Let’s take a look at the source code
3. Run the application

Q1: What is the difference between hitting the home button (HOME) and back button (ESC) ?

Q2: What is Log.v and how can it be used?