Today’s Topics

- Android Introduction
- Building your first app!

What is Android?

- 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 the more recent ART runtime) 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

- **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

- 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

- 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 / ART run time.
  - Documentations
  - Sample code
Android Studio: IDE

- Android Studio
  - New IDE from Google
  - Uses Gradle for building the apps and packages.

- There was no dedicated IDE for Android from Google prior to Android Studio
  - 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.
  - ADT Bundle
  - Eclipse with the ADT Plugin.

Application Framework

- 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
  - View System
    - 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

- 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
  - LibWebCore
    - 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

- 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 or ART runtime.
- The Dalvik VM executes files in the Dalvik Executable (.dex) format. Dalvik uses a Just-in-Time model (JIT).
- ART, a new runtime. It is compatible with DEX. ART uses Ahead-of-Time model (AOT) for improved performance.

Java

- 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 Bytecode & Virtual Machine

- 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

- 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

Passing Arguments

- 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

- Keyword `extends` to inherit from a superclass.
- Example
  ```java
  package edu.kettering.hellokettering;

  import android.os.Bundle;

  public class HelloKettering extends AppCompatActivity {
      /** Called when the activity is first created. */
      @Override
      public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);
      }
  }
  ```
Developing for Mobile Apps

Design Considerations

- 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

- Being fast and efficient
  - [http://developer.android.com/training/articles/perf-tips.html](http://developer.android.com/training/articles/perf-tips.html)
  - Avoid creating short-term 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

Application Not Responding (ANR)

- Activity Manager and Window Manager monitor application responsiveness.
- Android displays the ANR dialog when it detects one of the 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

- Your application can cause problems under the multitasking environment when you ignore seamlessness issues.
- Be a good citizen!
  - Save instance state
  - Keep in mind that Android is a mobile platform.
  - Another app can pop up anytime 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

Building Your First App

Creating an Android Project

- Application Name
  - App name: appears to users.
- Project Name
  - Name for your project directory.
- Package Name
  - Must be unique across all packages on the Android system.
  - For this, use the reverse domain name of your organization.
- Minimum Required SDK
  - The lowest version of Android that your app supports.
- Target SDK
  - The highest version of Android with which you have tested with your app.

Directories and Files

- AndroidManifest.xml
  - It describes the fundamental characteristics of the app.
- src/
  - Your app’s main source files.
- res/
  - Contains several sub dirs.
    - drawable-hdpi/
    - Drawable objects (bitmaps) for high-density screen.
  - layout/
    - Files that define your app’s user interface
  - values/
    - XML files that contains a collection of resources

Kettering University
Using a Real Device

- Enable USB debugging on your device.
  - Settings > Applications > Development for Android 3.2 or older
  - Settings > Developer options for Android 4.0 or newer.
  - This menu item is hidden.
- Settings > About phone and find Build number item. Tap it seven times.

Using the Emulator

- Android Virtual Device (AVD) must be created.
  - Window > Android Virtual Device Manager.
  - Device Definitions tab.
  - Choose Nexus 4 by Google.
  - Click Create AVD…

View and ViewGroup

- View objects are UI widgets such as buttons and text fields.
- ViewGroup objects are *invisible* view containers.
  - Define how the child views are laid out. (e.g. grid or vertical list).
- Open activity_main.xml from res/layout/

Layout

- Relative/Linear/Absolute/Table/Grid/Frame
- We will cover these in a lecture later.

- Also see the more recent construct called Fragments.
  - Modular design enables easier composition of multiple View-like objects for large screens.

Building a User Interface
Add a Text Field

- Place the Text Field at below of textView1.
- Take a peek in activity_main.xml
  - id: a unique id for the view.
  - The @ sign is required to refer to any resource object from XML
  - The + sign to define a resource id. The id will be automatically generated by the SDK tools.
  - layout_width/height: size of the view
  - wrap_content: as big as needed to fit the contents of the view.
  - layout_alignLeft/below: relative layout.
  - layout_marginTop: top margin.
  - em: a unit of width in the filed the typography.

Add a Button

- Change the Text, “Button” to “Send.”
- Using a String resource
  - res/values/strings.xml
  - Add “button_send” with “Send” as its value.

Respond to the Send Button

- Add “android:onClick attribute to the Button and set its attribute “sendMessage”
- Open the MainActivity class (src/)
- Add this corresponding method.
  ```java
  /** Called when the user clicks the Send button */
  public void sendMessage(View view) {
      // Do something in response to button
  }
  ```
  This generates an error:
  - Automatic Building!
  - Suggests possible fixes.
  - Select “import View”
  - import android.view.View; is automatically added.

Accessing to a View

- A special class R!
  - All resource IDs are defined in your project’s R class.
  - The R class is generated by the SDK tool.
- Accessing to a resource:
  - In XML: @type/id
  - In code: R.type.id
  - Example) a string resource hello (@string/hello) can be accessed by the name R.string.hello.
- Add this code to the sendmessage method.
  ```java
  EditText editText1 = (EditText)findViewById(R.id.editText1);
  TextView textView1 = (TextView)findViewById(R.id.textView1);
  textView1.setText(editText1.getText());
  ```
Questions?