**CMSC 436**

Handheld device characteristics:

* Small, portable, battery-powered
* Accept touch, audio, light, and radio input
* Uses very different inputs
* Networked
  + Wifi, cellular, and bluetooth
* Truly personal
  + Rarely shared
* Limited resources
  + CPU, memory, battery, I/O bandwidth
* Limited screen space
  + Traditional webpages have assumed a large screen
* Diverse Usage Context
  + Indoor vs. Outdoor
  + Light and climate
  + Private vs. Public settings
* Handheld devices do different things in different ways
  + Highly context-dependent
  + Bursty usage patterns – from seconds to minutes
* Limited attention
  + Mobile usage IMPLIES multitasking

Introduction to the Android Platform:

* A software stack for mobile devices:
  + OS kernel, system libraries, application frameworks and key apps
* Android Software Development Kit (SDK) for creating apps
  + Libraries and development tools
  + Lots of documentation
* Android Architecture
  + Linux Kernel 🡪 Libraries 🡪 Android Runtime 🡪 Application Framework 🡪 Applications
  + Linux Kernel
    - In general
      * Lowest level of the handheld device
      * Security
      * Memory and process management
      * File and network I/O
      * Device driver
    - Android specific
      * Power management
      * Android shared memory
      * Low memory killer (because the mobile device is memory constrained, android actively kills less used apps)
      * Interprocess communication (IPC)
      * And more!
  + Libraries
    - System C library
      * Bionic libc – android’s own memory efficient version of C
    - Surface Manager
      * Display management
    - Media Framework
      * Audio/Video
    - Webkit
      * Browser engine
    - OpenGL
      * Graphics engines
    - SQLite
      * Relational
      * Database engine
  + Android Runtime
    - Two main components
      * Core Java libraries
        + Basic java classes – java.\*, javax.\*

Not *exactly* the same as the Java JDK

* + - * Runtime
        + Dalvik VM
* Typical Workflow
  + App written in java
  + Compile to java bytecode files
  + *dx* converts java bytecode files to a single dex bytecode file (classes.dex)
  + Virtual machine executes dex bytecode file

Apps are executed by a virtual machine

Designed for resource constrained environments

Slower CPU

Less RAM

Limited battery life

* + - * + Android Runtime (ART)

Compiles dex bytecode to native binary

Applies system-dependent optimizations

Done at installation time

Results in faster execution at cost of larger executable

* + Application Framework
    - Package Manager
      * Keeps track of app packages on device
    - View System
    - Resource Manager
      * Manages non-compiled resources (e.g., strings graphics, etc.)
    - Activity Manager
      * Manages app lifecycle and navigation stack
    - ContentProvider
      * Inter-application data sharing
        + Ex. contact list between apps like Facebook and Twitter
    - Location Manager
      * Provides location and movement information
    - Notification Manager
  + Applications
    - Standard apps
      * Home screen, browser, phone, etc.
      * Nothing special about these apps
        + You can substitute your own or 3rd party app

The Android Emulator

* Pros
  + Doesn’t require an actual phone
  + Hardware is reconfigurable
  + Changes are non-destructive
* Cons
  + Can be very slow
  + Some features are unavailable
    - No support for Bluetooth or USB connection
  + The AVD is not the device, performance may be misleading
* Advanced Options
  + Can emulate different device/user characteristics such as
    - Network speed/latencies
    - Battery power
    - Location coordinates

To connect to AVD:

* *NameOfApp user*$ telnet localhost *value*

Debugger

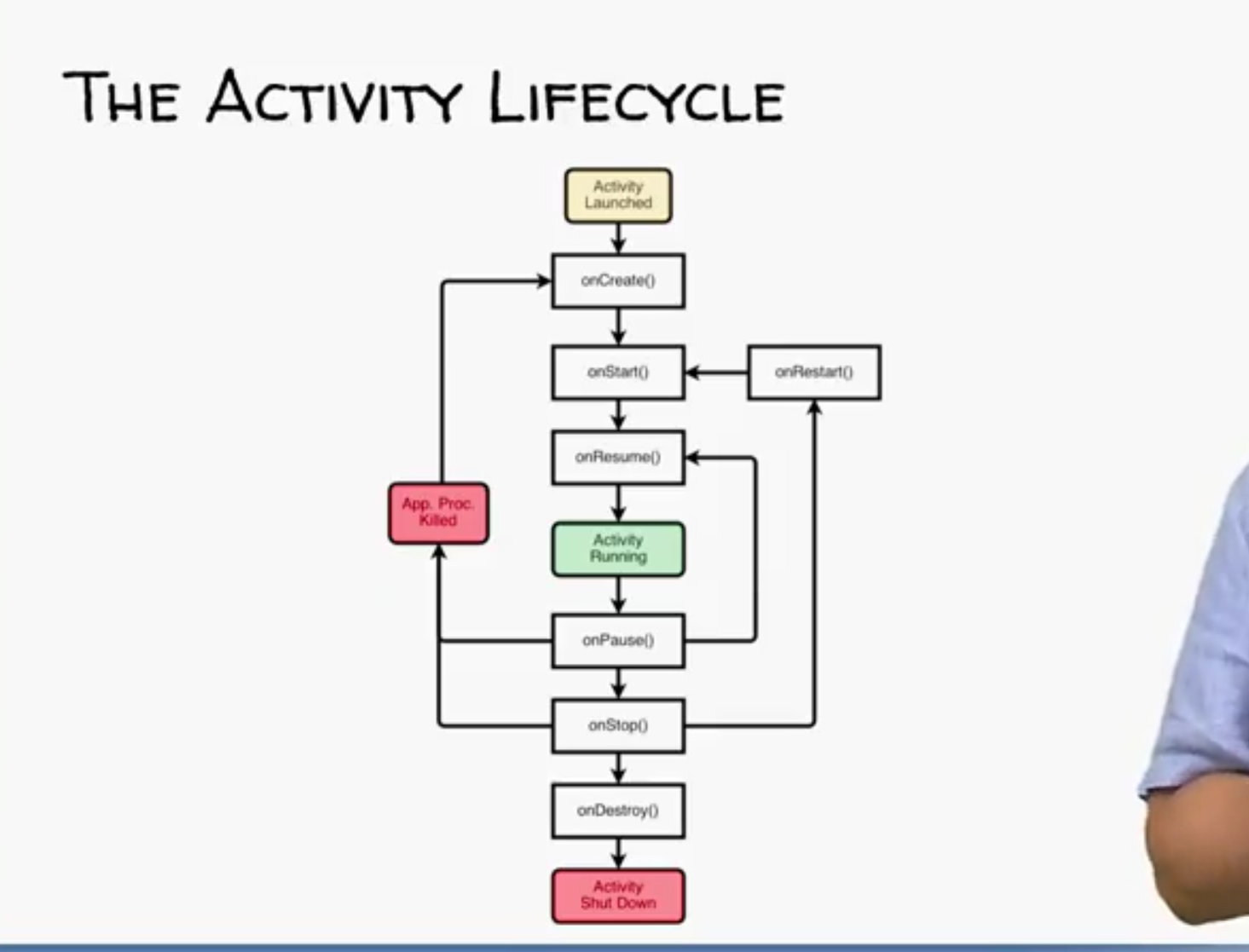
* CTRL + D to debug
* Dalvik Debug Monitor Service (DDMS)
  + General tools for monitoring device behavior
    - Traceview
      * Trace methods in application calls
      * Uses traditional debugger
      * CTRL + 6
      * Holy shit this is cool
    - UI Hierarchy viewer
      * Sees the tree of interface stuff

Application Fundamentals:

* Components:
  + Activity
    - Primary class for user interaction
    - Should support a single, focused task (dialing a phone number, etc.)
  + Services
    - In the background operations (ex. music players)
  + Broadcast receivers
    - Listen for and respond to events (ex. messaging app)
      * Subscriber in the publisher/subscriber pattern
      * Events represented by the intent class and then broadcast
      * Broadcast receiver receives and then responds to event
  + Content providers
    - Allows multiple applications to store and share data (ex. browser app)
      * Database style interface
      * Handles interprocess communication
  + Map location
* Building an application:
  + Android Project 🡪 Compilation and Packaging 🡪 Android Package 🡪 Signing 🡪 Device/Simulator
  + Steps:
    - Defining resources
      * Resources are non-source code entities
      * Allows apps to be customized for different devices and users
      * Ex. Strings
        + String
        + String array
        + Plurals
      * Ex. User interface layout
        + Usually written in in .xml
        + Can specify different layout files
    - Implementing application classes
      * Usually involves creating one activity using onCreate()
      * onCreate() workflow:
        + Restore saved state
        + Set content view
        + Initialize UI elements
        + Link UD elements to code actions
    - Package application
      * System packages application components and resources into a .apk file
      * Developers specify required application information in a file called androidmanifest.xml
        + Contains name of application, list of components, permission, hardware features, minimum API level
    - Install and run application

The Activity Class

* Provide a visual, interactive interface
* Should be modular, should support a single, focused thing
* Applications often comprise of several activities
* Navigation through activities:
  + Tasks
    - A task is a set of related activities
    - These related activities don’t have to be part of the same application
    - Most tasks start at the home screen
  + The task backstack
    - When an activity is launched, put on the top of the backstack
    - When activity is destroyed, activity is popped of task backstack
  + Suspended and resumed activities
    - Activities are created, destroyed, suspended, and resumed
    - Some of these actions depend on user behavior
    - Some depend on Android
* Activity Life Cycle States
  + Resumed/Running – visible, user interacting
  + Paused – visible, user not interacting, can be terminated\*
  + Stopped - not visible, can be terminated
* Lifecycle Methods:
  + onCreate()
  + onStart()
  + onDestroy()
  + etc.



Activity visibility between onResume() and onPause()

OnCreate()

* Call super.onCreate()
* Set the Activity’s ContentView
* Retain references to UI view as necessary
* Configure views as necessary

The Intent Class

* Specifies an operation to be performed
* Intent Fields
  + Action – names operation to be performed
  + Data – represents data associated with the intent formatted with URI
  + Category – denotes type that can handle intent
  + Minetype – Specifies MIME type of data
  + Component – Identify intent’s target activity
  + Extras – store additional information, map of key-value pairs
  + Flags -
* Starting Activity with Intents
  + Explicit Activation
  + Implicit Activation via Intent Resolution