Android

Android Application Development

- Ashwin

Agenda

- ***Android Platform Overview***
- ***Installation***
- ***Building Blocks***
- ***Application Development***
- ***Development Tools***
- ***Source walk through***

Introduction to Android

- *Open software platform for mobile development*
- *A complete stack – OS, Middleware, Applications*
- *An Open Handset Alliance (OHA) project*
- *Powered by Linux operating system*
- *Fast application development in Java*
- *Open source under the Apache 2 license*

Installation

System requirements

Download

http://ftp.cs.pu.edu.tw/pub/eclipse/eclipse/downloads/drops/R-3.7-201106131736/eclipse-SDK-3.7-linux-gtk.tar.gz

http://dl.google.com/android/ADT-12.0.0.zip

http://dl.google.com/android/android-sdk_r12-linux_x86.tgz

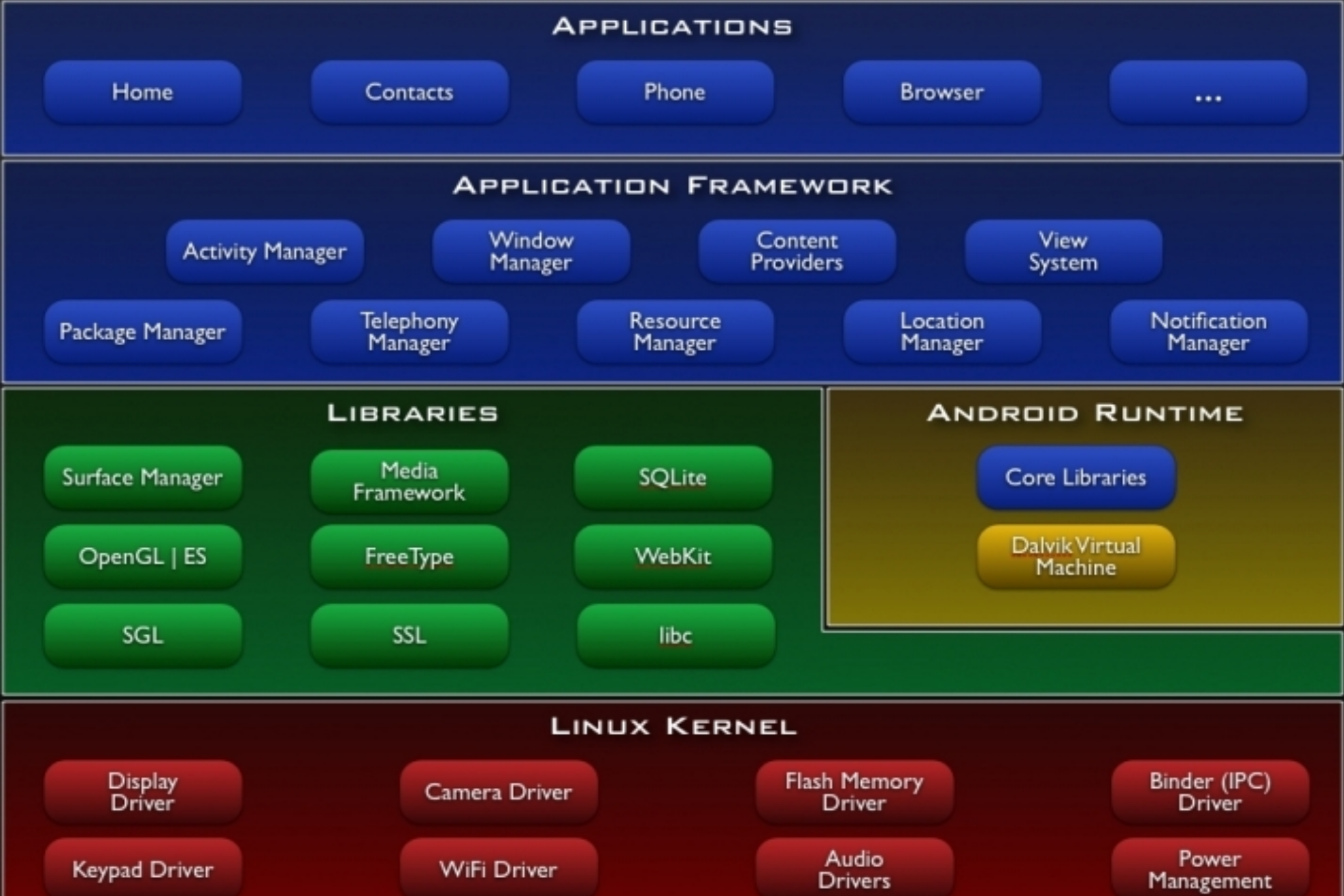
Installing

Eclipse

Untar eclipse-SDK-3.7-linux-gtk.tar.gz .

Open Handset Alliance

- *What is the Open Handset Alliance (OHA)? (1)*
- **It's a consortium of several companies**


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
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
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

Opportunities

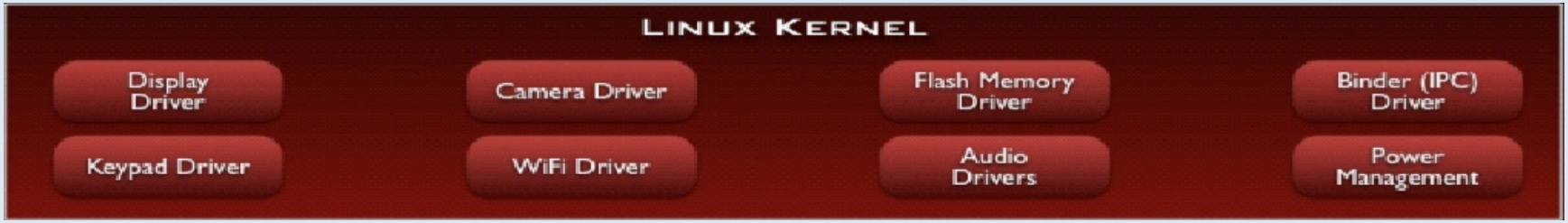
- ***100s of millions of mobile phone users***
- ***Very high growth, esp. in Asia***
- ***think 1985 on the desktop (if you were born by then)***
- ***no dominant 3rd party developers.... yet***
- ***what will the killer app categories be?***
- ***what does it mean to have any app + the internet in your pocket?***
- ***You can develop for it today!***

Android Versions

- ***Cupcake***
- ***Donut***
- ***Eclair***
- ***Froyo***
- ***Gingerbread***
- ***Honeycomb***
- ***Ice Cream Sandwich***
- ***Jelly Bean***

Linux Kernel

- ***Works as a HAL***
- ***Device drivers***
- ***Memory management***
- ***Process management***
- ***Networking***

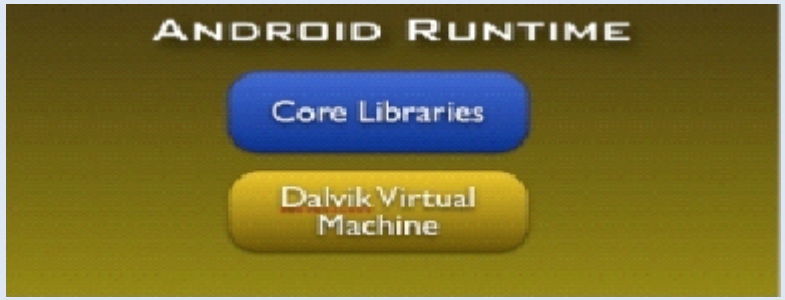


Libraries

- *C/C++ libraries*
- *Interface through Java*
- *Surface manager – Handling UI Windows*
- *2D and 3D graphics*
- *Media codecs, SQLite, Browser engine*

Android Run Time

- ***Dalvik VM***
  - Dex files
  - Compact and efficient than class files
  - Limited memory and battery power
- ***Core Libraries***
  - Java 5 Std edition
  - Collections, I/O etc…

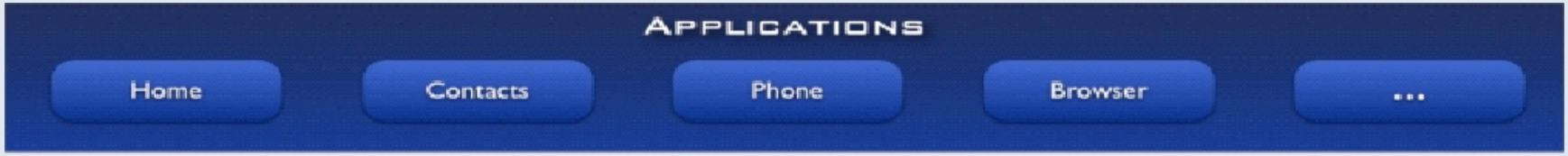


Application Framework

- ***API interface***
- ***Activity manager – manages application life cycle.***
- ***Package Manager***

Applications

- ***Built in and user apps***
- ***Can replace built in apps***



Development Tools

- ***Eclipse***
- ***Android SDK***
- ***3rd Party Add On's***

Emulator

- ***QEMU-based ARM emulator***
- ***Runs the same image as the device***

- ***Limitations:***
  - No Camera support

Installation

- ***http://developer.android.com/sdk/installing/index.html***
- ***http://onthefencedevelopment.com/blog/installing-eclipse-and-and***
- ***http://sharepointkunskap.wordpress.com/2012/05/11/install-android-sdk-and-eclipse-in-ubuntu-12-04/***
- ***http://kittipatkampa.wordpress.com/2012/01/17/how-to-install-eclipse-and-android-sdk-on-ubuntu-10-04-lts/***
- ***http://www.wikihow.com/Install-Android-on-Ubuntu-Linux-With-Eclipse-Ide***

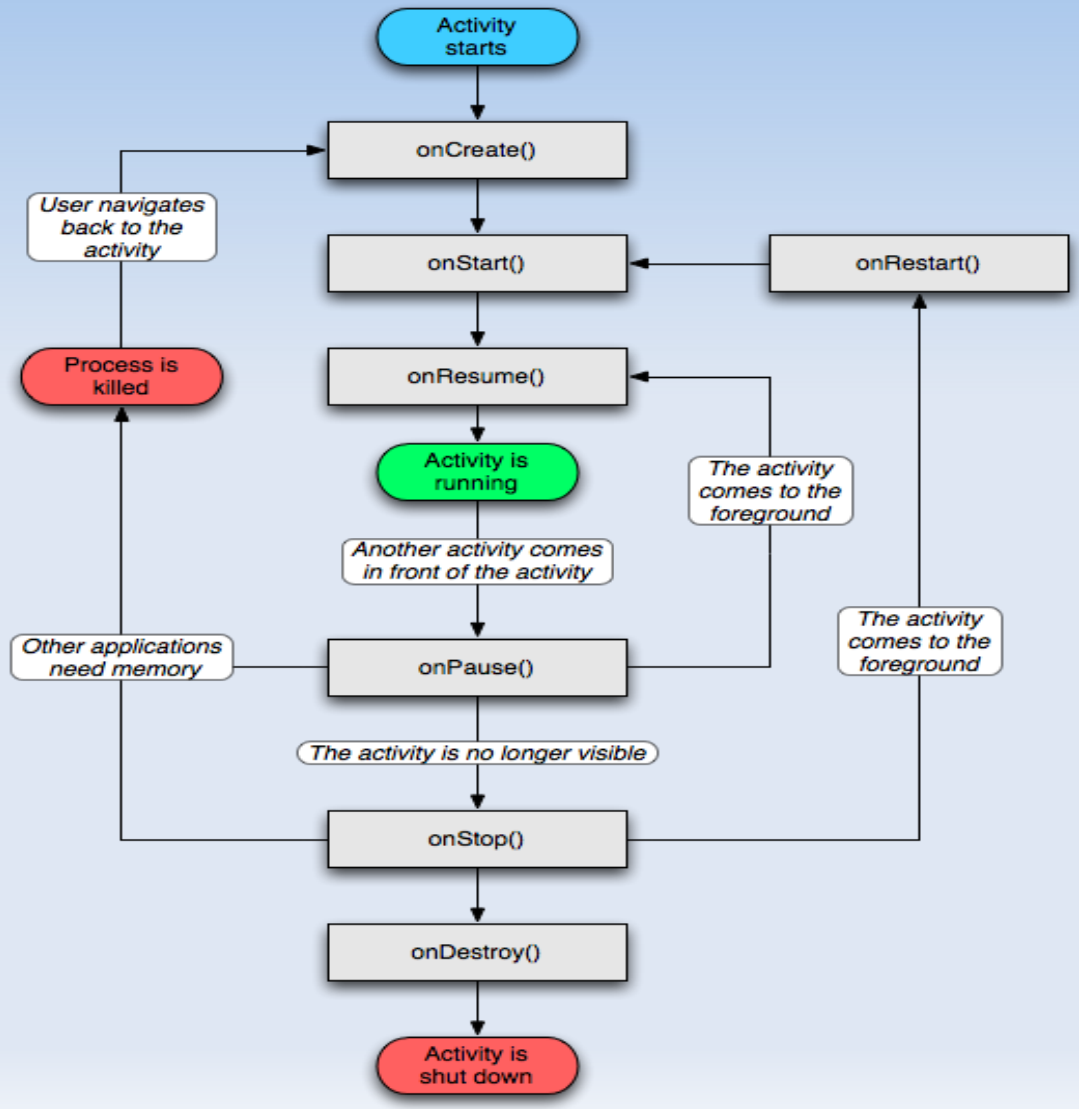
Building Blocks

- ***Activity***
- ***Service***
- ***Intent and IntentFilters***
- ***Content Providers***
- ***Processes and Threads***

Activity

- *Typically correspond to one UI screen*
- *But, they can:*
  - *Be faceless*
  - *Be in a floating window*
  - *Return a value*

Activity Life Cycle



Activity Life Cycle

- ***OnCreate***
- ***OnStart***
- ***OnResume***
- ***OnPause***
- ***OnStop***
- ***OnDestroy***

Application Lifecycle

- ***Application run in their own processes (VM, PID)***
- ***Processes are started and stopped as needed to run an application's components***
- ***Processes may be killed to reclaim resources***

Security

- ***Each application runs in its own process***
- ***Process permissions are enforced at user and group IDs assigned to processes***
- ***Finer grained permissions are then granted (revoked) per operations***

UI Overview

- ***All user interface elements in an Android app are built using View and ViewGroup objects.***

- ***A View is an object that draws something on the screen that the user can interact with.***

- ***A ViewGroup is an object that holds other View (and ViewGroup) objects in order to define the layout of the interface.***

Introducing Layouts

- ***Layout is the architecture for the user interface in an Activity.It defines the layout structure and holds all the elements that appear to the user.***
- ***You can declare your layout in two ways:***
  - ***Declare UI elements in XML***
  - ***Instantiate layout elements at runtime***

Layouts

- ***Linear Layout: A view group that aligns all children in a single direction, vertically or horizontally. You can specify the layout direction with the android:orientation attribute.***
- ***Relative Layout:RelativeLayout is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements.***

Layouts

- ***FrameLayout: The simplest of the Layout Managers, the Frame Layout simply pins each child view to the top left corner. Adding multiple children stacks each new child on top of the previ-ous, with each new View obscuring the last.***
- ***GridLayout: A layout that places its children in a rectangular grid.***
- ***TableLayout: A layout that arranges its children into rows and columns.***

Layout Classes

- ***AbsoluteLayout In an Absolute Layout, each child View's position is defined in absolute coor-dinates. Using this class, you can guarantee the exact layout of your components, but at a price.***

View

- ***This class represents the basic building block for user interface components.***
- ***Provides classes that expose basic user interface classes that handle screen layout and interaction with the user.***
- ***View is the base class for widgets, which are used to create interactive UI components (buttons, text fields, etc.)***

Input Control

- ***Button : A push-button that can be pressed, or clicked, by the user to perform an action.***
- ***Text field : An editable text field. You can use the AutoCompleteTextView widget to create a text entry widget that provides auto-complete suggestions***
- ***Checkbox : An on/off switch that can be toggled by the user. You should use checkboxes when presenting users with a group of selectable options that are not mutually exclusive.***
- ***Radio button : Similar to checkboxes, except that only one option can be selected in the group.***
- ***Toggle button : An on/off button with a light indicator.***

Input Control

- ***Spinner: A drop-down list that allows users to select one value from a set.***
- ***Pickers : A dialog for users to select a single value for a set by using up/down buttons or via a swipe gesture. Use a DatePickercode> widget to enter the values for the date (month, day, year) or a TimePicker widget to enter the values for a time (hour, minute, AM/PM), which will be formatted automatically for the user's locale.***
- ***ImageView***
- ***ProgressBar***

List View

- ***A view that shows items in a vertically scrolling list.***

- ***The items come from the ListAdapter associated with this view.***

Adapter

- *An Adapter object acts as a bridge between an AdapterView and the underlying data for that view.*
- *The Adapter provides access to the data items.*
- *AdapterView objects have two main responsibilities:*
  - ** Filling the layout with data*
  - ** Handling user selections*

Grid View

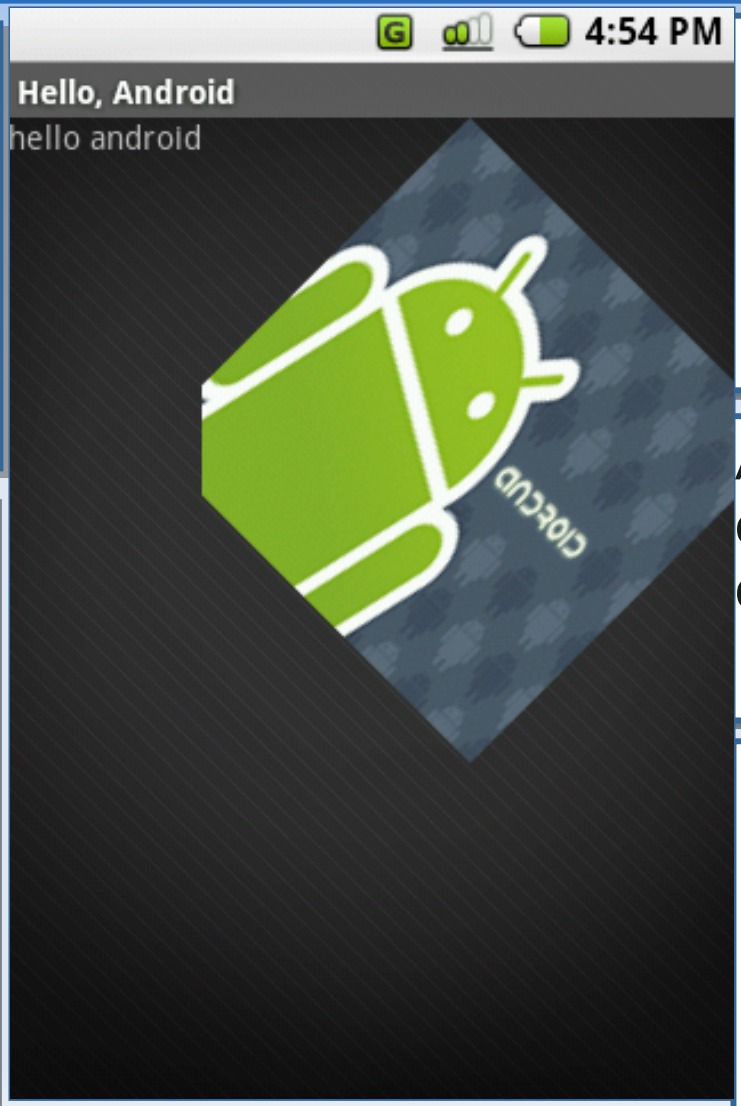
- ***GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.***

- ***The grid items are automatically inserted to the layout using a ListAdapter.***

Applications have common structure

Views such as lists, grids, text boxes, buttons, and even an embeddable web browser

Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data



An **Activity Manager** that manages the life cycle of applications and provides a common navigation backstack

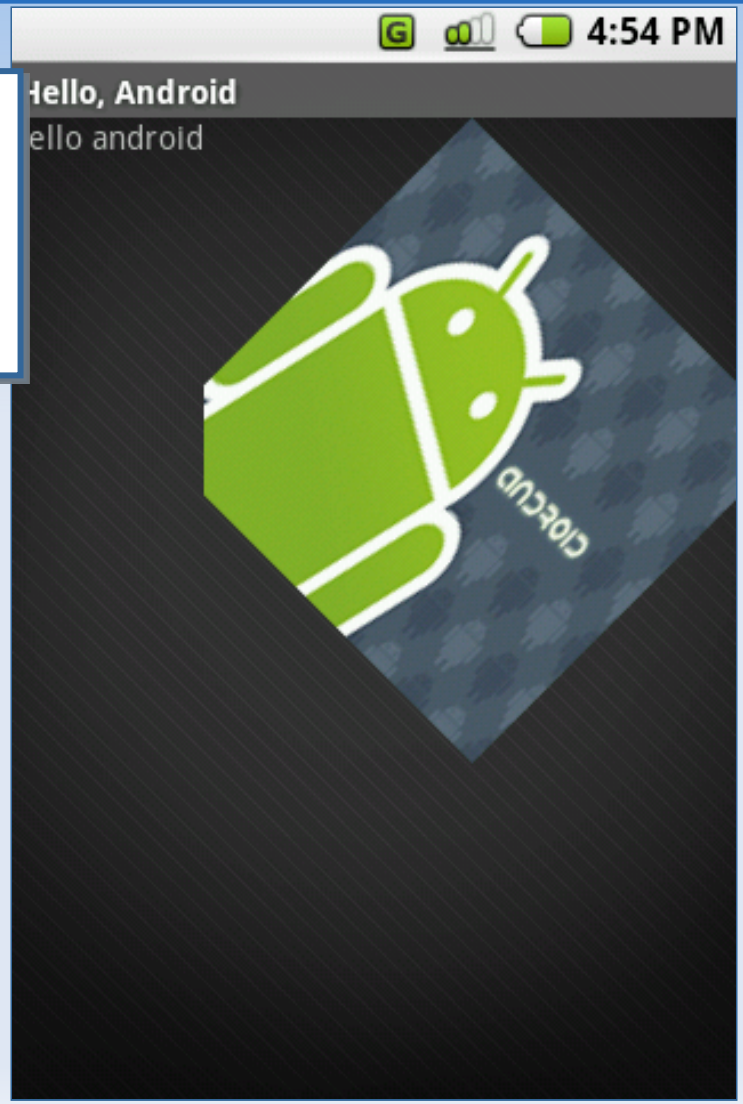
A **Notification Manager** that enables all apps to display custom alerts in the status bar

A **Resource Manager**, providing access to non-code resources such as localized strings, graphics, and layout files

Applications have common structure

Broadcast receivers can trigger intents that start an application

Data storage provide data for your apps, and can be shared between apps – database, file, and shared preferences (hash map) used by group of applications

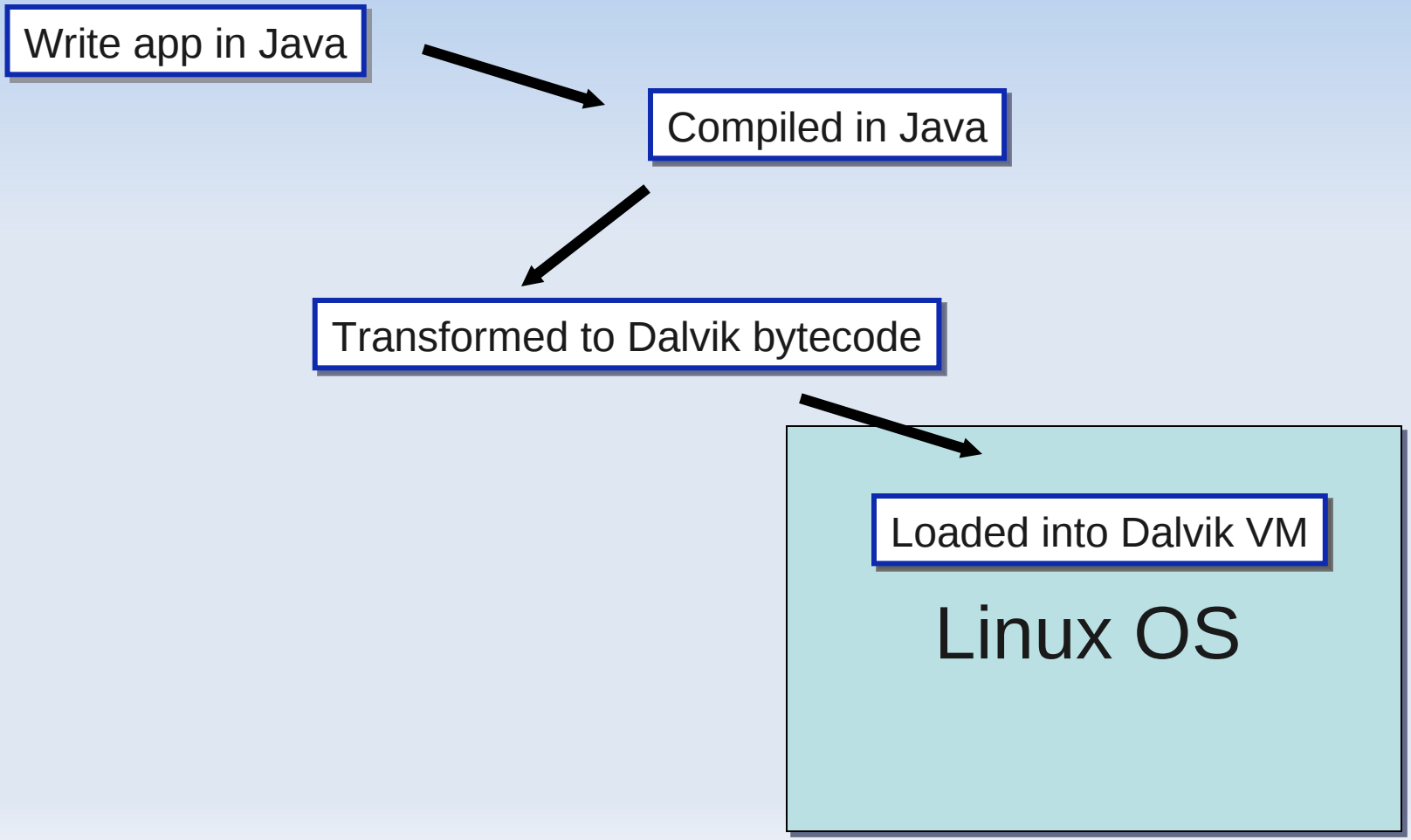


Activity is the presentation layer of your app: there will be one per screen, and the Views provide the UI to the activity

Intents specify what specific action should be performed

Services run in the background and have no UI for the user – they will update data, and trigger events

Compilation



Dalvik VM

Run multiple VMs efficiently

Each app has its own VM

Minimal memory footprint

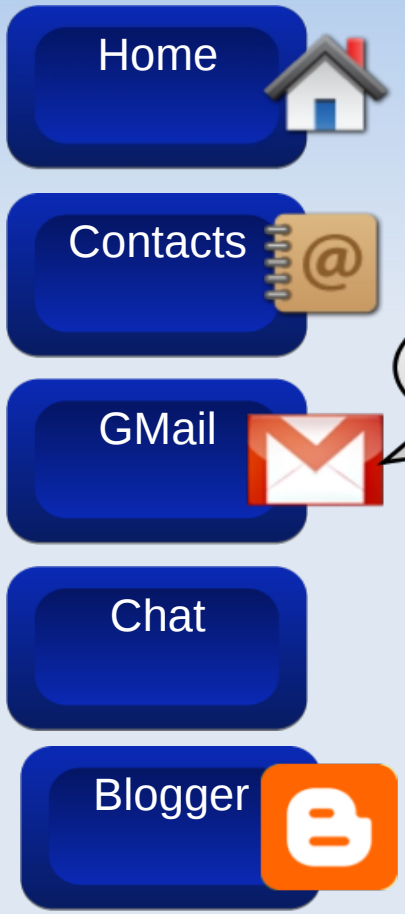
Hardware

- ***Turn on "USB Debugging" on your device.***
  - ***Settings > Applications > Development and enable USB debugging***
- ***Set up your system to detect your device. e.***
  - ***Log in as root and create this file: /etc/udev/rules.d/51-android.rules.***
  - ***SUBSYSTEM=="usb", ATTR{idVendor}=="0bb4", MODE="0666", GROUP="plugdev"***

Intents

- ***Think of Intents as a verb and object; a description of what you want done***
  - ***E.g. VIEW, CALL, PLAY etc..***
- ***System matches Intent with Activity that can best provide the service***
- ***Activities and IntentReceivers describe what Intents they can service***

Intents



Client component makes a request for a specific action

System picks best component for that action

New components can provide existing functionality

Intent Receivers

- ***Components that respond to broadcast 'Intents'***
- ***Way to respond to external notification or alarms***
- ***Apps can invent and broadcast their own Intent***

Services

- ***Faceless components that run in the background***
  - ***E.g. music player, network download etc…***

Service Life Cycle

- ***Bound And UnBound Service***
- ***A facility for the application to tell the system about something it wants to be doing in the background***
- ***A bound service allows components (such as activities) to bind to the service, send requests, receive responses, and even perform interprocess communication (IPC).***

Shared Preferences

- ***A class provides a general framework that allows you to save and retrieve persistent key-value pairs of primitive data types.***

- ***Can save any primitive data: booleans, floats, ints, longs, and strings. This data will persist across user sessions (even if your application is killed).***

Content Providers

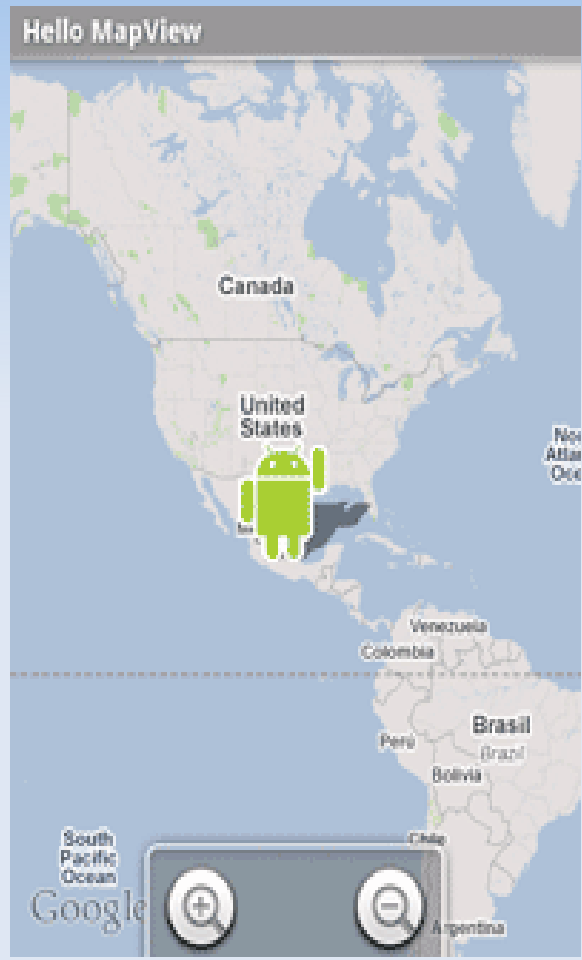
- ***Enables sharing of data across applications***
  - ***E.g. address book, photo gallery***
- ***Provides uniform APIs for:***
  - ***querying***
  - ***delete, update and insert.***
- ***Content is represented by URI and MIME type***

Notifications

- ***Several types of situations may arise that require you to notify the user about an event that occurs in your application.***
  - ***When an event such as saving a file is complete, a small message should appear to confirm that the save was successful.***
  - ***If the application is performing work that the user must wait for (such as loading a file), the application should show a hovering progress wheel or bar.***

Location Manager

- ***This class provides access to the system location services.***



Hardware Oriented Features

Feature	Description
Camera	A class that enables your application to interact with the camera to snap a photo, acquire images for a preview screen, and modify parameters used to govern how the camera operates.
Sensor	Class representing a sensor. Use getSensorList(int) to get the list of available Sensors.
SensorManager	A class that permits access to the sensors available within the Android platform.
SensorEventListener	An interface used for receiving notifications from the SensorManager when sensor values have changed. An application implements this interface to monitor one or more sensors available in the hardware.
SensorEvent	This class represents a sensor event and holds information such as the sensor type (e.g., accelerometer, orientation, etc.), the time-stamp, accuracy and of course the sensor's data.
MediaRecorder	A class, used to record media samples, that can be useful for recording audio activity within a specific location (such as a baby nursery). Audio clippings can also be analyzed for identification purposes in an access-control or security application. For example, it could be helpful to open the door to your time-share with your voice, rather than having to meet with the realtor to get a key.
GeomagneticField	This class is used to estimated estimate magnetic field at a given point on Earth, and in particular, to compute the magnetic declination from true north.
FaceDetector	A class that permits basic recognition of a person's face as contained in a bitmap. Using this as a device lock means no more passwords to remember — biometrics capability on a cell phone.

Getting Started

- ***http://developer.android.com/index.html***
- ***Blog http://android-developers.blogspot.com/ which has lots of useful examples***
- ***http://www.anddev.org***