Introduction to Android Development

JamesHarmon@gmail.com

Course Overview

- This is a hands-on course that covers Android Development
- Introduction to Android development for developers with no Android experience
- Logistics
  - Hours
  - Breaks
  - Lunch

Course Objectives

- Understand the important concepts in Android development
- Be able to use the tools for developing Android apps
- Be able to develop and pubish your own Android app

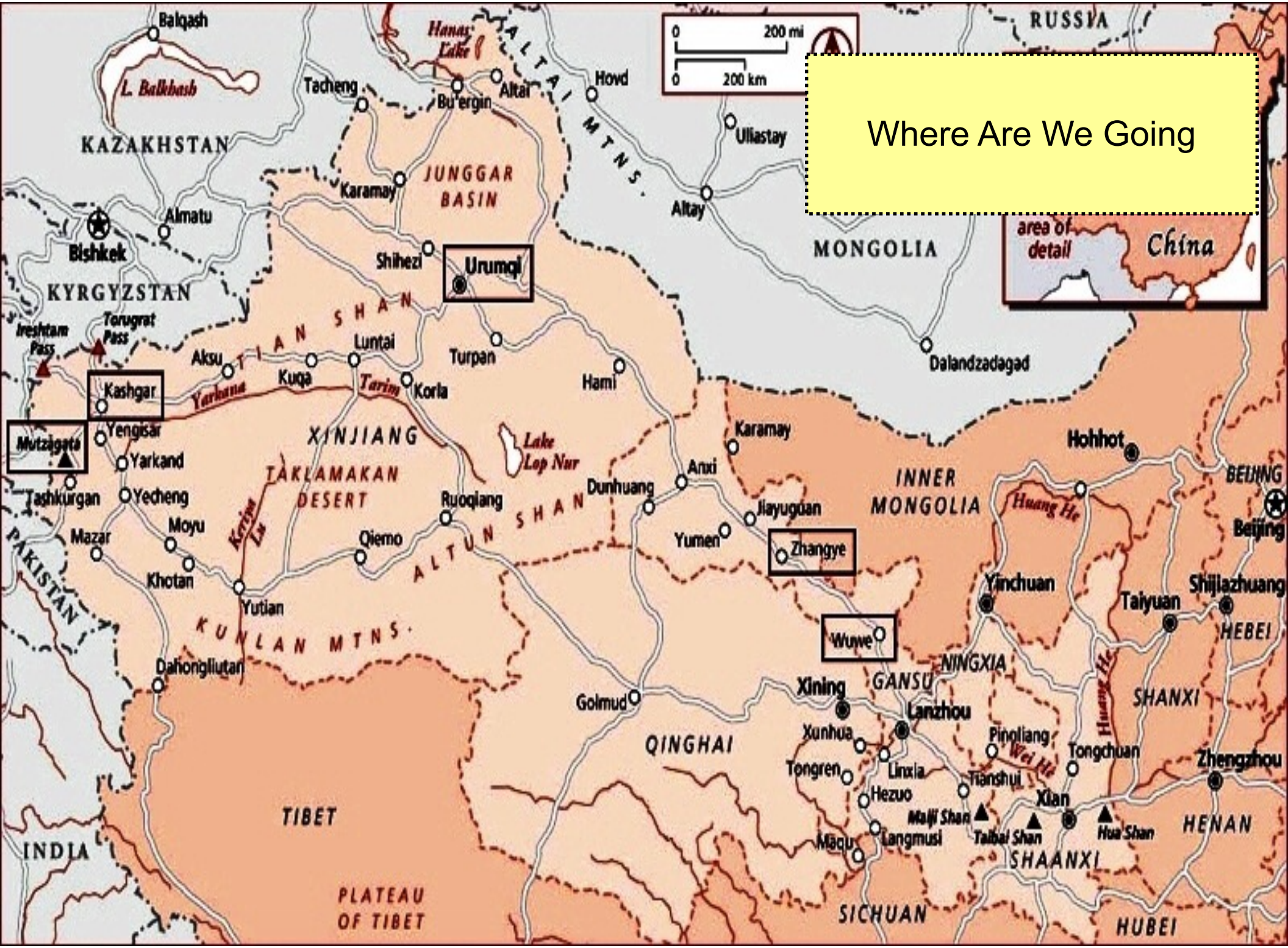
Target Audience and Prerequisites

- This course is for people that want to learn how to develop Android applications
  - Java developers
  - Managers who want to know concepts and terminology

- Course prerequisites:
  - Knowledge of Java, SQL, and relational database concepts
  - Knowledge of web development
  - Knowledge of mobile development

Course Agenda

- See Outline


Where Are We Going
Balqash
L. Balkhash
Hanas Lake
Tacheng
Bu'ergin
Altai
ALTAI MTNS.
Hovd
Uliastay
RUSSIA
0 200 mi
0 200 km
KAZAKHSTAN
Karamay
JUNGGAR BASIN
Altay
MONGOLIA
area of detail
China
Almatu
Bishkek
KYRGYZSTAN
Shihezi
Urumqi
Dalandzadagad
Ireshtam Pass
Torugrat Pass
TIAN SHAN
Luntai
Aksu
Turpan
Kuqa
Tarim
Korla
Hami
Kashgar
Yarkand
Karamay
Hohhot
Mutzagata
Yengisar
XINJIANG
Lake Lop Nur
Yarkand
TAKLAMAKAN DESERT
Anxi
INNER MONGOLIA
Huang He
BEIJING
Beijing
Tashkurgan
Yecheng
Ruoqiang
Dunhuang
Jiayuguan
Mazar
Moyu
Keriya He
Qiemo
Yumen
Zhangye
PAKISTAN
ALTUN SHAN
Yinchuan
Shijiazhuang
Taiyuan
Khotan
Yutian
HEBEI
Wuwe
KUNLAN MTNS.
NINGXIA
Dahongliutan
Xining
GANSU
SHANXI
Golmud
Lanzhou
Xunhua
Pinoliang
Tongchuan
QINGHAI
Tongren
Linxia
Wei He
Zhengzhou
Tianshui
Hezuo
Xian
TIBET
Maji Shan
Hua Shan
HENAN
INDIA
Maqu
Langmusi
Taibai Shan
SHAANXI
PLATEAU OF TIBET
SICHUAN
HUBEI

Where have I been

- Java Developer
- Android User
- Android Developer
- Android Instructor

Instructor Information

- James Harmon - JamesHarmon@gmail.com
- Senior Instructor, Accelebrate
  - Android Training
  - Java and J2EE training
- Project Manager, Accenture
  - Large Scale DataBase Systems
- B.S. in Computer Science
  - University of Illinois at Urbana-Champaign
- Author - Using the Dojo JS Framework

Student Introduction

- Name and Department
- Describe your experience with Java
- Describe your experience with IntelliJ
- Describe your experience with Mobile development
- Describe your experience with Android development
- What do you want to get out of this class?

Section 01
The Basics

What is Android
Building Apps

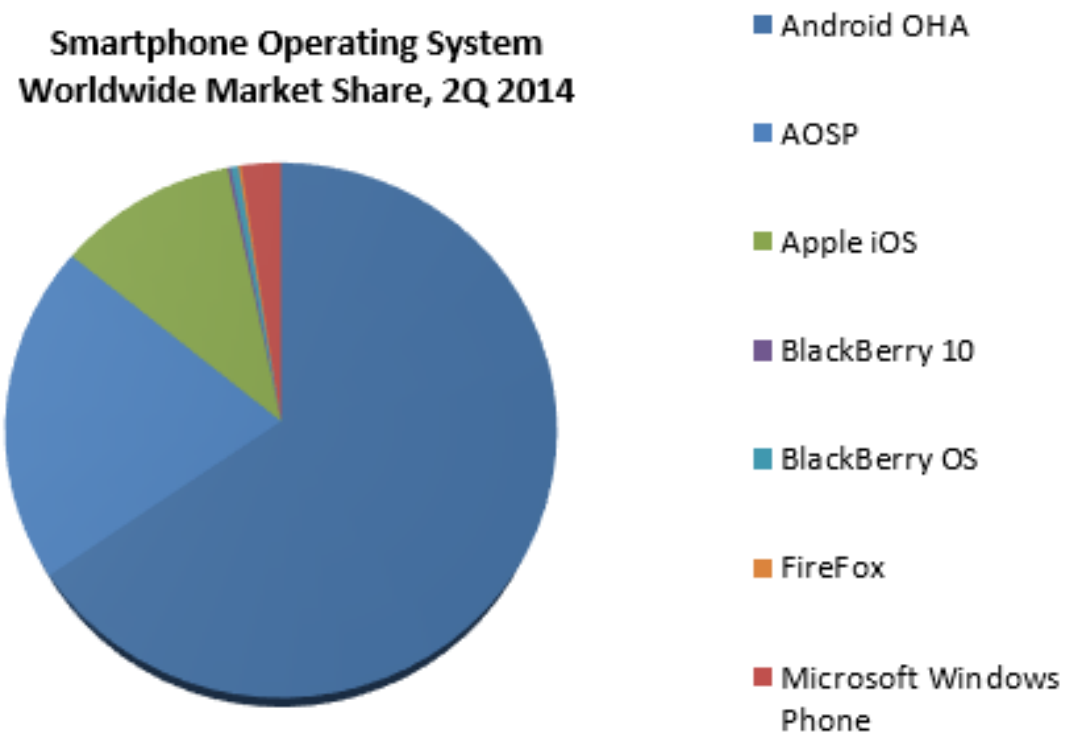
Session Objectives

- Discuss Android platform
- Explain Android App Development

What is Android?

- Android is a Linux-based operating system
- Designed primarily for touchscreen mobile devices such as smartphones and tablet computers
- Developed by Google in conjunction with the Open Handset Alliance
  - a consortium of 86 hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices
- Initially developed by Android Inc, whom Google financially backed and later purchased in 2005
- Android was announced by Google in 2007

Android Market Share – 85%

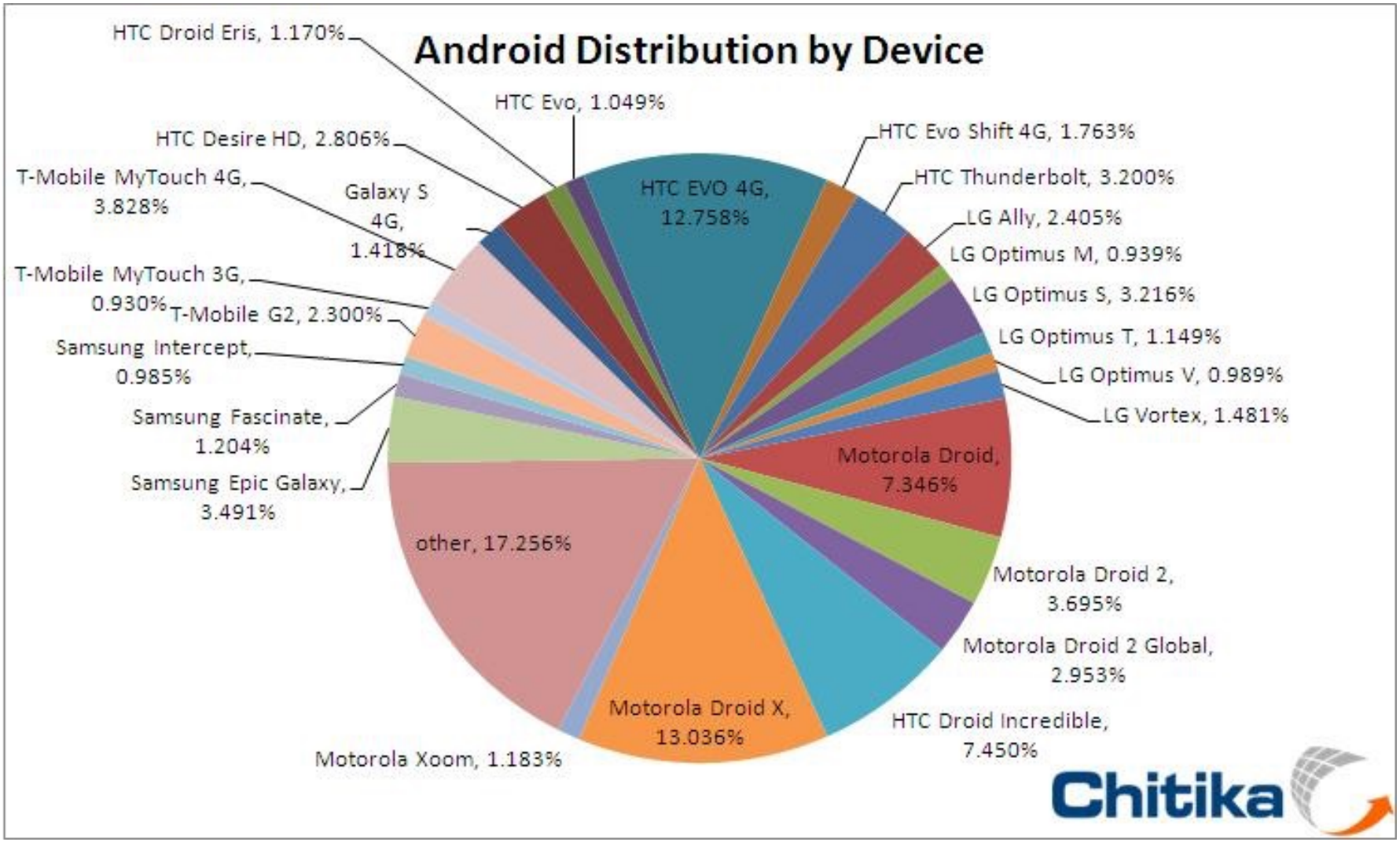


"Recent numbers from ABI Research on the market share of mobile smartphone platforms splits out the two major variants of Android. Both Google's flavor of Android (namely the Android variant used by members of the Open Handset Alliance, with the Google Play support and services), and the Android Open Source Project, which is free for any manufacturer to base their handset on, are listed.
Google's preferred version of Android is on 65%, while the Android Open Source Project (AOSP) is on 20%, comfortably ahead of iOS and Windows Phone."

Source: Forbes Magazine

http://www.forbes.com/sites/ewanspence/2014/08/05/how-google-benefits-from-the-increased-market-share-of-the-android-open-source-project/

A Multitude of Devices



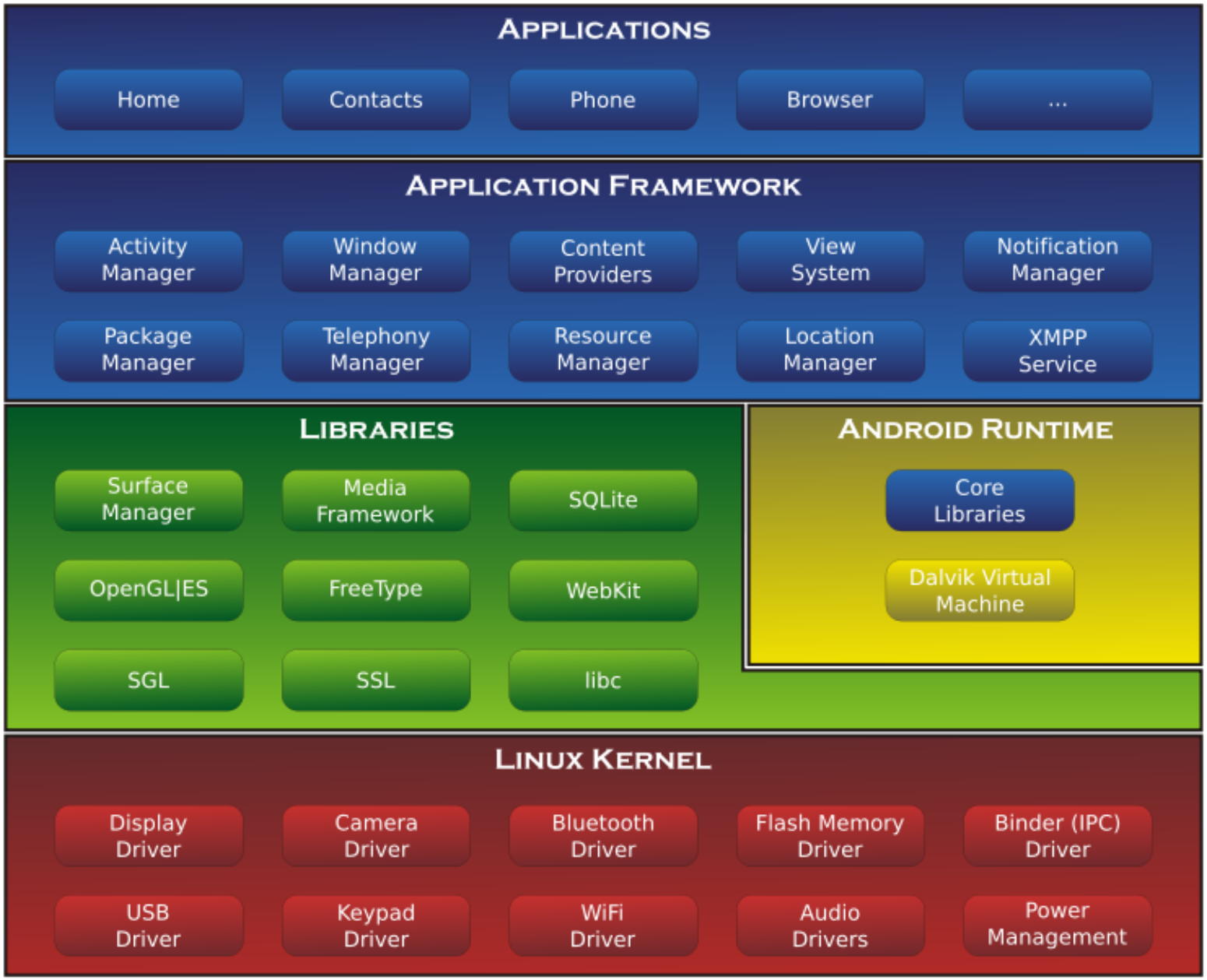
Other Uses of Android

- Past
  - Tablet Computers
  - Google Glass
- New
  - Android TV
  - Android Wear (Watches)
- And who knows what's next?
  - Laptops (Windows Replacement)
  - Cars
  - Home Automation
  - Specialized Devices

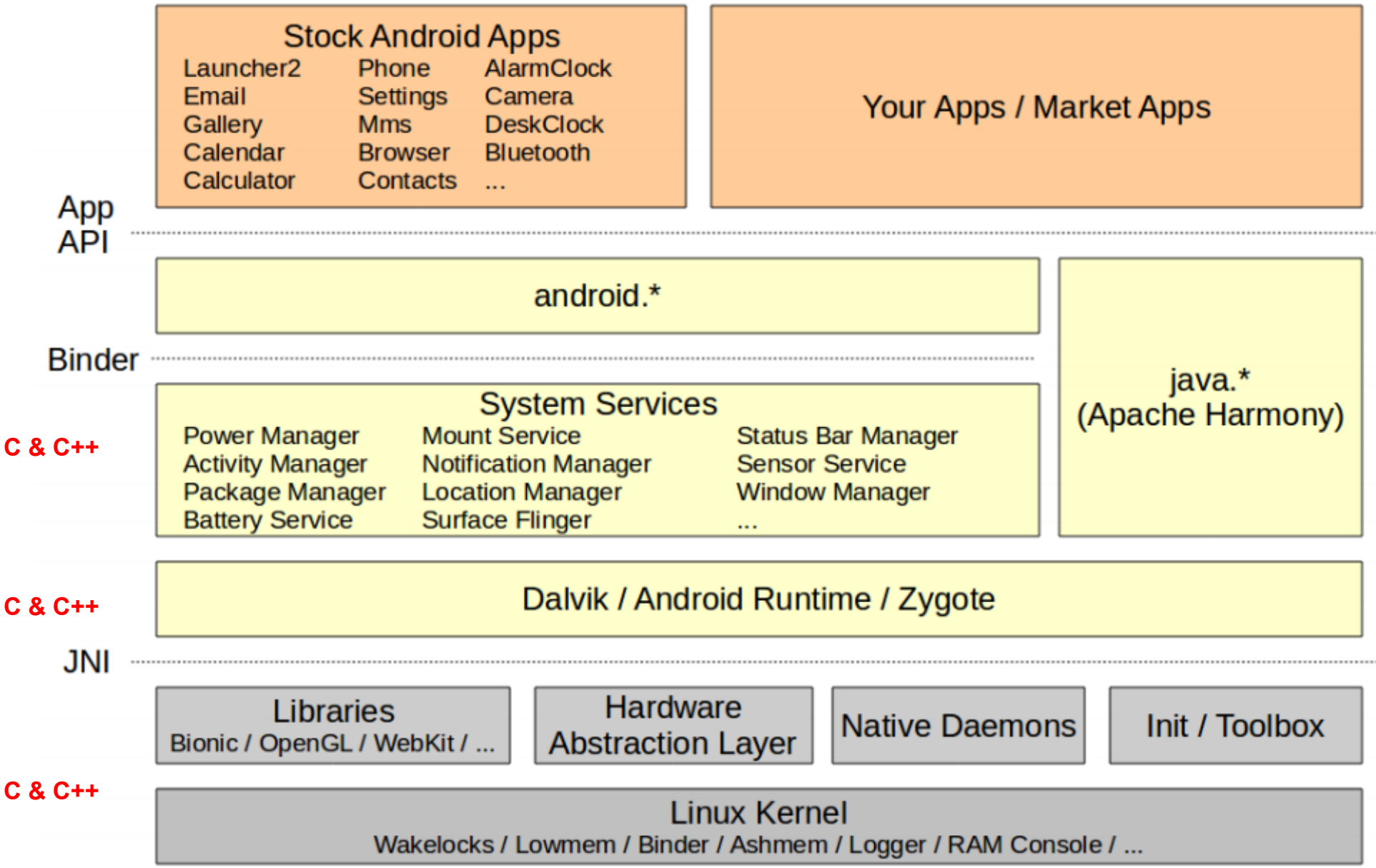
Android Hardware

- No absolute requirements
- Touch Screen
- Vibrate Haptics
- Bluetooth
- WiFi
- GPS
- NFC
- Flash / RAM / SD Card
- Sensors
  - Accelerometer
  - Magnatometer
  - Thermometer

What is Android – The Deep Dive



What is Android – The Deep Dive II

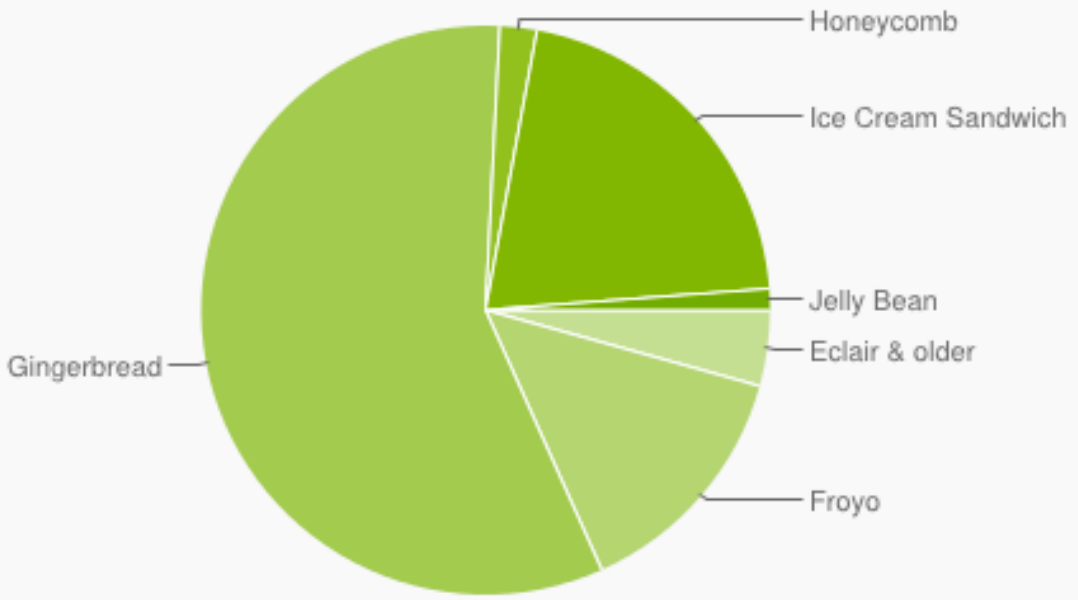


Linux Kernel

- Wakelocks
  - Mechanism to force device to stay on
  - Built on top of Linux power management functionality
- Lowmem (low memory killer)
  - Activated before Linux kernel Out-Of-Memory killer
  - Kills Java processes when low memory is detected
- Binder
  - RPC/IPC mechanism (OpenBinder)
  - Controls all interprocess communication
- Ashmem – Anonymous Shared Memory
  - Replacement for POSIX SHM functionality
- Alarm
  - On top of Linux HRT (High Res Timer)
  - Wake device up regardless of suspend status

Multiple Versions – Then (2 years ago)

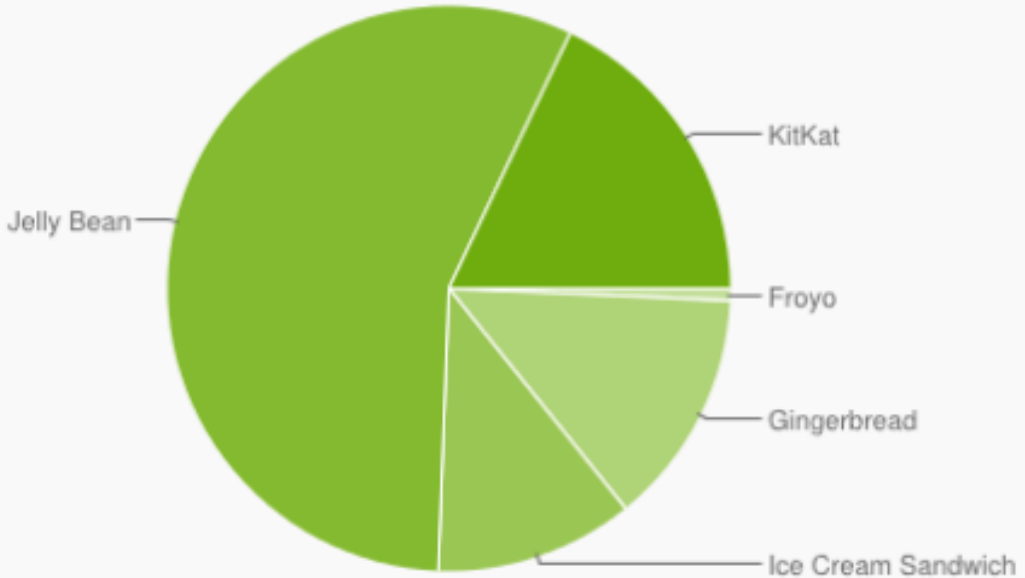
Version	Codename	API	Distribution
1.5	Cupcake	3	0.2%
1.6	Donut	4	0.4%
2.1	Eclair	7	3.7%
2.2	Froyo	8	14%
2.3 - 2.3.2	Gingerbread	9	0.3%
2.3.3 - 2.3.7		10	57.2%
3.1	Honeycomb	12	0.5%
3.2		13	1.6%
4.0 - 4.0.2	Ice Cream Sandwich	14	0.1%
4.0.3 - 4.0.4		15	20.8%
4.1	Jelly Bean	16	1.2%



Data collected during a 14-day period ending on September 4, 2012

Multiple Versions - Now

Version	Codename	API	Distribution
2.2	Froyo	8	0.7%
2.3.3 - 2.3.7	Gingerbread	10	13.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	11.4%
4.1.x	Jelly Bean	16	27.8%
4.2.x		17	19.7%
4.3		18	9.0%
4.4	KitKat	19	17.9%



Data collected during a 7-day period ending on July 7, 2014.
Any versions with less than 0.1% distribution are not shown.

Android Milestones

- October 2003
  - Android Inc founded by Andy Rubin
- August 2005
  - Google acquires Android Inc
- November 2007
  - Open Handset Allianced Formed – Android Introduced
  - Android Beta SDK released
- September 2008
  - Android 1.0 Released – featured on HTC Dream (G1)
- October 2009
  - Android 2.0 released
- December 2010
  - Android 2.3 Released

Android Milestones

- February 2011
  - Android 3.0 Released
- October 2011
  - Android 4.0 Released
- May 2012
  - Oracle sues Google for copyright infringement of Android Java framework – Google wins – APIs not copyrightable
- July 2012
  - Android 4.1 (Jelly Bean) Released
- December 2013
  - Andy Rubin replaced by Sundar Pichai
- May 2014
  - Oracle v. Google – reverses decision – back to district court to determine if “fair use” applies

Andy Rubin

- Carl Zeiss AG, robotics engineer, 1986–1989.
- Apple Inc., manufacturing engineer, 1989–1992.
- General Magic, engineer, 1992–1995. An Apple spin-off where he participated in developing Magic Cap, an operating system and interface for hand-held mobile devices.
- MSN TV, engineer, 1995–1999. When Magic Cap failed, Rubin joined Artemis Research, founded by Steve Perlman, which became WebTV and was eventually acquired by Microsoft.
- Danger Inc., co-founder, 1999–2003. Founded with Matt Hershenson and Joe Britt. Firm is most notable for the Danger Hiptop, often branded as the T-Mobile Sidekick, which is a phone with PDA-like abilities. Firm was later acquired by Microsoft in February 2008.
- Android Inc., co-founder 2003–2005.
- Google, 2005–present. Senior Vice President in charge of Android for most of his tenure. Since December 2013, managing the robotics division of Google (which includes companies bought by Google, such as Boston Dynamics).

Sundar Pichai – Google VP

- Responsible for
  - Chrome
  - Google Apps
  - Android

Anatomy of an Android App

- Activities
- Services
- Content Providers
- Broadcast Receivers
- Intents
- Application Manifest

Activities

- Single screen with a user interface
- Independent but work together
- Can be invoked from other applications

- Extends the Activity class

Service

- Perform long-running operations in the background
- No User Interface
- Can bind to other services or activities

- Extends the Service class

Content Provider

- Manages shared application data
- Provides consistent interfaces to data
  - Restful
  - CRUD operations
- Data Store backing options
  - SQLite DB
  - File System
  - Web
- Can consume data from other Content Providers

- Extends the `ContentProvider` class

Broadcast Receiver

- Respond to system wide messages
- Messages can be initiated by system or another app
- 2 kinds
  - Delivered asynchronously to all receivers
  - Delivered in priority order Should be light weight, work should be passed to services
- Usually Short-Lived

- Extends the BroadcastReceiver class

Intents

- Messages that link app components together
  - Explicit – launch a specific component
  - Implicit – launch any component which is registered to handle intent

- Describes an operation to be performed
  - Requested action
  - Data used by the action
  - Extra metadata

Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>

<manifest>
    <uses-permission />

    <application>
        <activity> … </activity>
        <service> … </service>
        <provider> … </provider>
        <receiver> … </receiver>
    </application>

</manifest>
```

Android NDK

- NDK is a toolset that allows you to implement parts of your app using native-code languages such as C and C++.

- Allows reuse of existing code libraries written in these languages, but most apps do not need the Android NDK.

- NDK will not benefit most apps. As a developer, you need to balance its benefits against its drawbacks. Notably, using native code on Android generally does not result in a noticable performance improvement, but it always increases your app complexity. In general, you should only use the NDK if it is essential to your app—never because you simply prefer to program in C/C++.

- Typical good candidates for the NDK are CPU-intensive workloads such as game engines, signal processing, physics simulation, and so on. When examining whether or not you should develop in native code, think about your requirements and see if the Android framework APIs provide the functionality that you need.

Midtronics

- Custom hardware platform from Mitec
- Android ASOP – 4.0.3 Ice Cream Sandwich
- Enhancing some OS functionality
- Only using Android SDK (Not NDK)
- Over 50,000 LOC

Midtronics Framework

- Wifi and Bluetooth Manager
- Data Access Layer
- Activity Wizard
- Demonstration of App

Section Review

1. What is Android

2. What are the component of a simple Android App

Section 02
Android SDK

SDK
ADT

Session Objectives

- Understand the Android Software Development Kit (SDK)
- Understand the Android Development Toolkit Eclipse Plugin

Emulator

- Provides software only device for testing
- Can be used instead of a real device

Writing an Android App

- Use Eclipse
- Use the Android Development Toolkit plugin for Eclipse
  - Create an Android Project
- Write Java Code
- Write XML for configuration and layout
- Deploy the Android project
  - Emulator
  - Device

Speeding Up The Emulator

- Make the screen size smaller
- Upgrade device RAM to 1Gig from 256MB
- Enable caching on virtual device
- Check the “boot device from snapshot” option to reduce startup time once a boot image has been created
- After you have a good snapshot, uncheck “create snapshot” or it will be very slow to shutdown because it will create a new snapshot each time the emulator is shutdown

- Use Intel Hardware Accelerated Execution Manager

Lab 2.1
Creating A Simple Android App

Section 03
Unit Testing and Debugging

Unit Testing
Logging
JavaDoc
Source Code
Lint

Session Objectives

- Understand how to performing Unit Testing and TDD in the Android environment

- Learn Android specific techniques for debugging
  - Logging
  - JavaDoc
  - Source Code
  - Lint for static code analysis

Unit Testing and TDD

- DVM is not the JVM
- Can’t use standard jUnit on DVM
- Android provides jUnit subclasses for the DVM which provide instrumentation

Philosophy of Debugging

- Programming is debugging
- Finding the solution to any programming error becomes trivial when examined with the right tool at the right level of detail
- Fix broken windows
  - Resolve all the errors
  - Resolve as many of the warnings as possible

Logging

- Write log messages in code

```
private final String TAG = "EventArrayAdapter";

Log.d(TAG, "Position: " + position);
```

- View log messages
  - Use LogCat

- Source of log messages
  - View custom messages
  - View messages created by emulator
  - View messages created by device

JavaDoc and Source Code

- A developer using open source code who does not look at the code is no better than a developer who can't
- As a debugging tool it can be very useful to link to the Android source code and the related JavaDoc
- Android has not provided a consistent technique for linking in JavaDoc in early version of the SDK
- Use a 3rd party plugin to provide consistent linkage to source code and JavaDoc for all the SK versions

Using Lint

- Introduced in ADT 16
- Android Lint provides static code analysis for Android projects
- Analysis not limited to just code but also includes layout and manifest files

Lab 3.1
Unit Testing and Debugging

Section 04
Android User Interface

Views

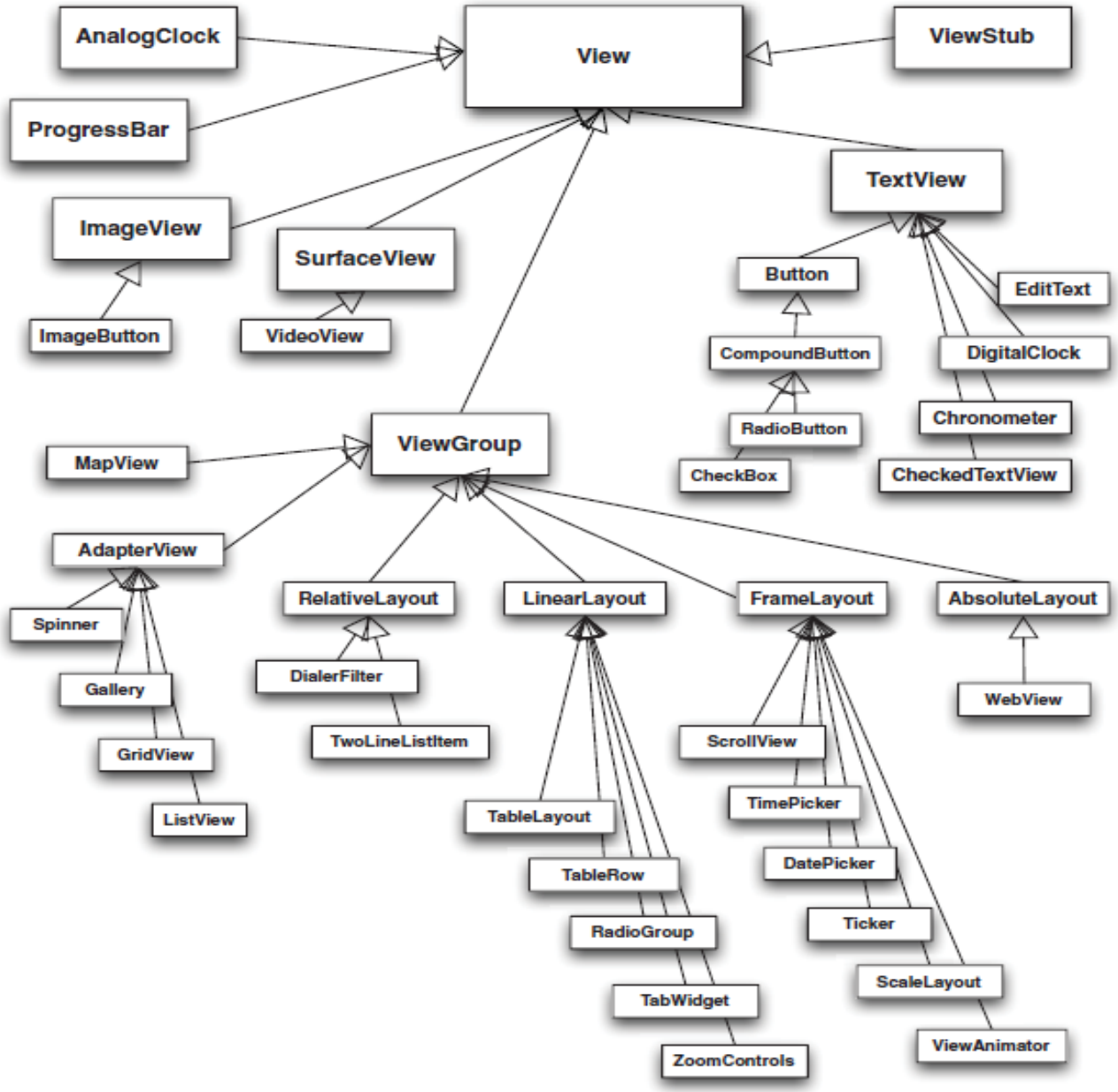
Session Objectives

- Views
- ListView
- Menus
- Activities

Views

- Provide the User Interface that is visible to the user
- Used by Activities
- Contained in layouts which provides a structured way to hold views

View Hierarchy



Methods In Base Class

Method	Purpose
`setBackgroundColor(int color)`	Set the background color.
`setBackgroundDrawable(Drawable d)`	Set the background `Drawable` (image).
`setClickable(boolean c)`	Set whether element is clickable.
`setFocusable(boolean f)`	Set whether element is focusable.
`setLayoutParams(ViewGroup.LayoutParams l)`	Set the `LayoutParams` (position, size, and more).
`setMinimumHeight(int minHeight)`	Set the minimum height (parent can override).
`setMinimumWidth(int minWidth)`	Set the minimum width (parent can override).
`setOnClickListener(OnClickListener l)`	Set listener to fire when click event occurs.
`setOnFocusChangeListener(OnFocusChangeListener l)`	Set listener to fire when focus event occurs.
`setPadding(int left, int right, int top, int bottom)`	Set the padding.

Additional Methods

Method	Purpose
`setGravity(int gravity)`	Set alignment gravity: top, bottom, left, right, and more.
`setHeight(int height)`	Set height dimension.
`setText(CharSequence text)`	Set text.
`setTypeFace(TypeFace face)`	Set typeface.
`setWidth(int width)`	Set width dimension.

Creating a layout

- Declarative (XML)
  - Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts.
- Programmatic (Java)
  - Instantiate layout elements at runtime. Your application can create View and ViewGroup objects (and manipulate their properties) programmatically.

XML Layout Editor

- WYSIWYG Editor for layouts
- Part of the ADT Plugin

Properties

- Set Property values
- Externalize values in string

Alternate layouts

- Provide different layouts
  - Orientation
  - Screen Size
  - Day or Night time
  - Locale

Section Review

1. What are Views
2. What are Layouts
3. How are layouts created

Section 04
Android User Interface

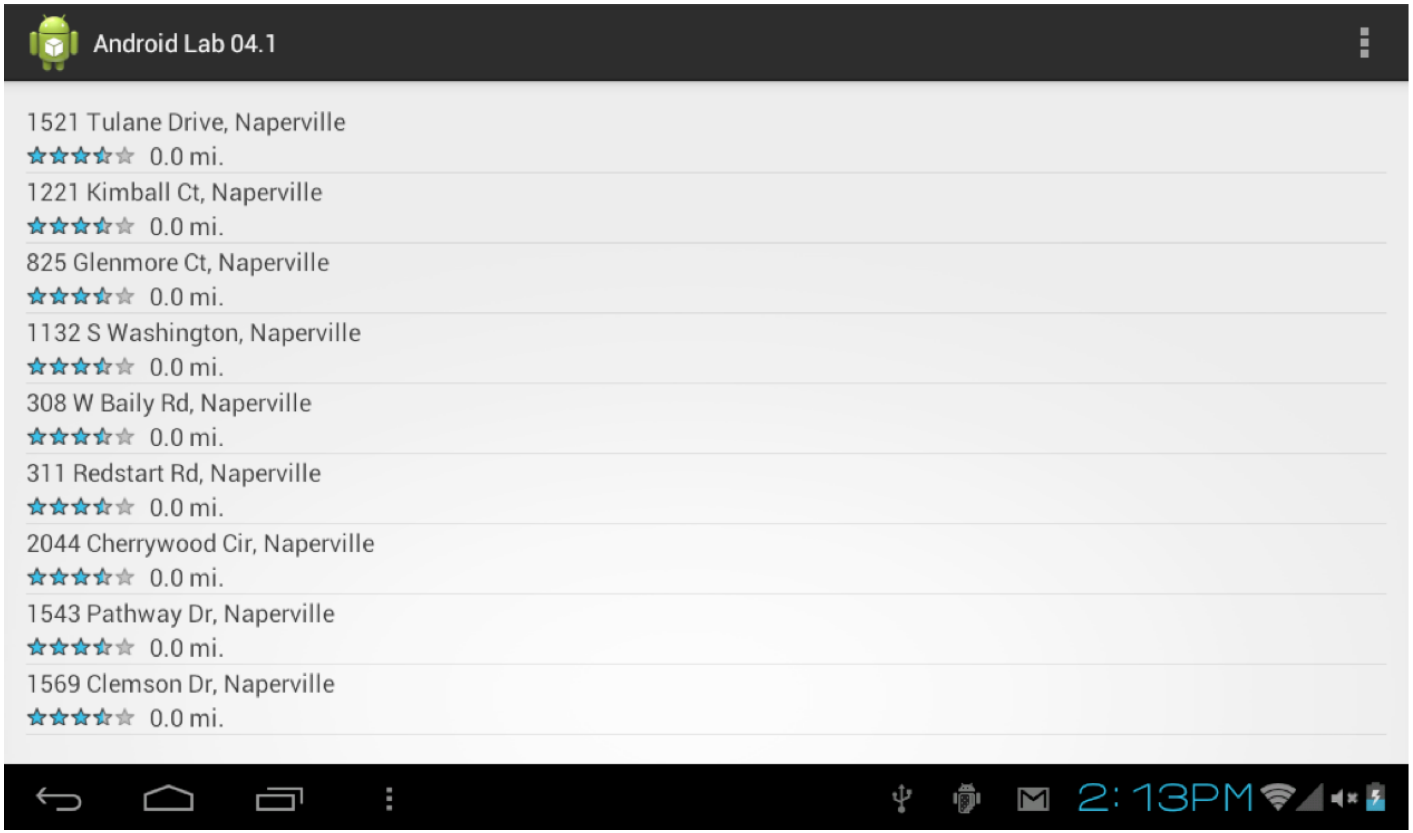
ListView

Objectives

- Add a simple ListView to the primary layout
- Create a custom adapter for displaying complex sale event information
- Optimize the list view using the View Holder pattern

List View

- One of the most common UI paradigms in mobile apps
- Displays a vertical, scrollable set of rows of view elements



List Views Are Virtual

- Problems displaying large data sets
  - Memory usage
  - Reduced performance because of constant object creation
- Solution
  - Populate rows on demand
  - Recycle views to reduce object churn

ListView – Top and Bottom

- Scrolling to the end of the ListView (or scrolling back to the top) causes the “blue cloud” to appear which designates that the list is at the end or the beginning

- Top/Bottom indicators vary by Android version

List View Adapter Can Use Any Data Source

- Adapters
  - Provide wrapper around any source of data
  - Provide a row view to the ListView
  - ListView never deals directly with the data

- GetView
  - Method implemented by Adapter to provide row view
  - Arguments
  - Returned object

Item Selection

- Rows can be selected by user by tapping
- Register a listener for the ListView (not for each row)

Creating a Simple ListView

- Declare a ListView element in a layout
- Get the data
- Build the adapter and supply the data
- Get a reference to the list view
- Attach the adapter
- Attach the listener

Declare a ListView Element in Layout

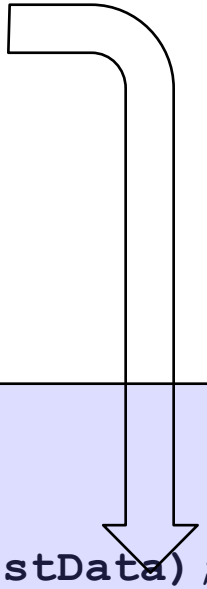
```
<RelativeLayout
    xmlns:android=http://schemas.android.com/apk/res/android
    android:layout_width="match_parent"
    android:layout_height="match_parent”>

    <ListView
        android:id="@+id/eventlistview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" >
    </ListView>

</RelativeLayout>
```

Acquire the Data (as a String Array)

```
List<Event> events = EventService.getAllEvents(this);

    List<String> listData = new ArrayList<String>();
    for (Event event : events) {
        listData.add(event.getStreet());
    }
```

Build the Adapter and Inject Data

```
List<Event> events = EventService.getAllEvents(this);

List<String> listData = new ArrayList<String>();
for (Event event : events) {
    listData.add(event.getStreet());
}

final ArrayAdapter<String> arrayAdapter =
    new ArrayAdapter<String>(
    this, android.R.layout.simple_list_item_1, listData);
```



Find A Reference to the ListView

```
List<Event> events = EventService.getAllEvents(this);

List<String> listData = new ArrayList<String>();
for (Event event : events) {
    listData.add(event.getStreet());
}

final ArrayAdapter<String> arrayAdapter =
        new ArrayAdapter<String>(
        this, android.R.layout.simple_list_item_1, listData);

ListView listView = (ListView)findViewById(R.id.eventlistview);
listView.setAdapter(arrayAdapter);
```

Attach the Adapter

```
List<Event> events = EventService.getAllEvents(this);

List<String> listData = new ArrayList<String>();
for (Event event : events) {
    listData.add(event.getStreet());
}

final ArrayAdapter<String> arrayAdapter =
        new ArrayAdapter<String>(
        this, android.R.layout.simple_list_item_1, listData);

ListView listView = (ListView)findViewById(R.id.eventlistview);
listView.setAdapter(arrayAdapter);
```

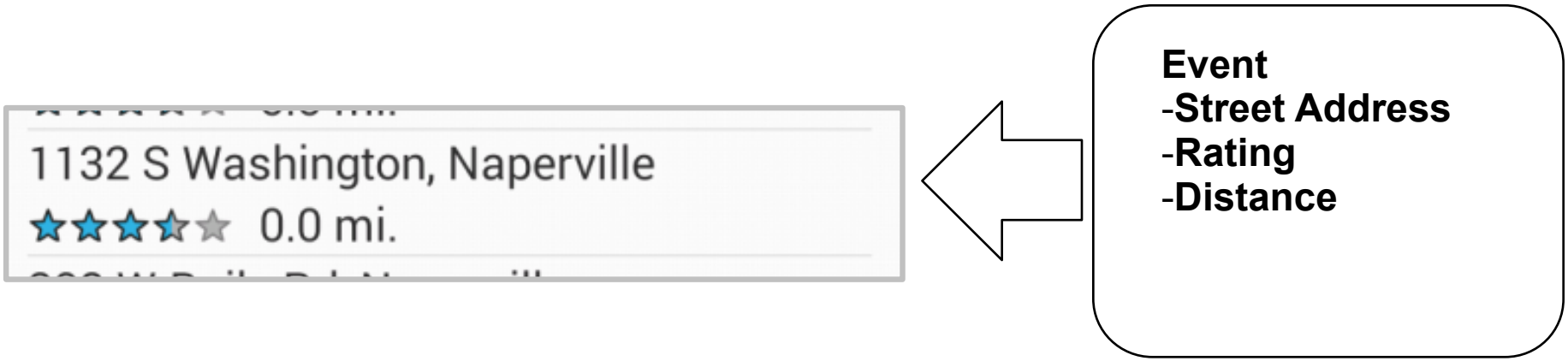
Attach The Listener

```
listView.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick
        (AdapterView<?> parent, View view, int position, long id) {
        Log.v(TAG, "User Clicked On ListView Position " + position);
    }
});
```

Lab 4.1 – Step 1
Create A Simple ListView

ArrayAdapter

- Provides binding between ListView, individual row and data
- Usually customized for complex row views
- Also customized for specific data types



Over-ride getView Adapter Method

View getView(int position, View convertView, ViewGroup parent)

- Position in the listview
- Recycled row view (may be null)
- The ListView itself

- returns a View

- Full data presentation control

How Does A ListView Work

- ListView creates enough convertView objects to display on the page
- ListView calls getView for each convertView object displayed
- As user scrolls list, ListView recycles existing convertView objects and replaces them with new data

Steps for Creating a Custom ArrayAdapter

- Declare the custom Array Adapter class and create the custom constructor
- Over-ride the getView method
  - Simple technique
  - Recycle technique
- Build the custom Array Adapter and Populate the Array Adapter with Data
- Assign the Array Adapter to the ListView

Declare Adapter and Constructor

```
public class EventArrayAdapter extends ArrayAdapter<Event> {

    public EventArrayAdapter
         (Context context, int resource, List<Event> events) {
        super(context, resource, events);
    }
```

Implement getView – The Simplest Way

```
public View getView(int position, View convertView, ViewGroup parent)
{
    View view= mInflater.inflate(R.layout.list_item_icon_text, null);

    TextView textField1 =
            (TextView) item.findViewById(R.id.textField1);
    textField1.setText("some text");

    return view;
}
```

Improve getView – Recycle Display Objects

```
public View getView(int position, View convertView, ViewGroup parent)
{
    if (convertView == null) {
        convertView = mInflater.inflate(R.layout.item, parent, false);
    }
    TextView textField1 =
            (TextView) item.findViewById(R.id.textField1);
    textField1.setText("some text");
    return convertView;
}
```

Build and Populate

```
List<Event> events = EventService.getAllEvents(this);

final ArrayAdapter<Event> arrayAdapter =
      new EventArrayAdapter(this, R.layout.event_list_item, events);
```

```
ListView listView = (ListView)findViewById(R.id.eventlistview);
listView.setAdapter(arrayAdapter);
```

Assign To ListView

```
List<Event> events = EventService.getAllEvents(this);

final ArrayAdapter<Event> arrayAdapter =
      new EventArrayAdapter(this, R.layout.event_list_item, events);
```

```
ListView listView = (ListView)findViewById(R.id.eventlistview);
listView.setAdapter(arrayAdapter);
```

Lab 4.1 – Step 2
Create A Custom ListView Adapter

ViewHolder Pattern

- Optimization to make ListView faster
- Maintains references to individual view elements within a complex row view
- Not an official type but a recommended best practice

Steps For Creating A View Holder

- Declare the View Holder
- Create the View Holder
- Assign the references for each child view
- Assign the values for the current position

Declare View Holder

```
static class ViewHolder {
    public TextView address;
    public RatingBar rating;
    public TextView distance;
}
```

Create the View Holder object

```
if (convertView == null) {
    LayoutInflater vi = (LayoutInflater) getContext().
         getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    convertView = vi.inflate(R.layout.event_list_item, null);

    ViewHolder viewHolder = new ViewHolder();

    viewHolder.address =
        (TextView) convertView.findViewById(R.id.item_address);
    viewHolder.rating =
        (RatingBar) convertView.findViewById(R.id.item_rating);
    viewHolder.distance = (TextView)
        convertView.findViewById(R.id.item_distance);

    convertView.setTag(viewHolder);
}
```

Assign The References to the Child Views

```
if (convertView == null) {
    LayoutInflater vi = (LayoutInflater) getContext().
         getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    convertView = vi.inflate(R.layout.event_list_item, null);

    ViewHolder viewHolder = new ViewHolder();

    viewHolder.address =
        (TextView) convertView.findViewById(R.id.item_address);
    viewHolder.rating =
        (RatingBar) convertView.findViewById(R.id.item_rating);
    viewHolder.distance = (TextView)
        convertView.findViewById(R.id.item_distance);

    convertView.setTag(viewHolder);
}
```

Save the View Holder with the convertView

```
if (convertView == null) {
    LayoutInflater vi = (LayoutInflater) getContext().
         getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    convertView = vi.inflate(R.layout.event_list_item, null);

    ViewHolder viewHolder = new ViewHolder();

    viewHolder.address =
        (TextView) convertView.findViewById(R.id.item_address);
    viewHolder.rating =
        (RatingBar) convertView.findViewById(R.id.item_rating);
    viewHolder.distance = (TextView)
        convertView.findViewById(R.id.item_distance);

    convertView.setTag(viewHolder);
}
```

Get a Reference to the ViewHolder for convertView

```
ViewHolder viewHolder = (ViewHolder) convertView.getTag();
```

```
viewHolder.address.setText(event.getStreet());
viewHolder.rating.setRating(event.getRating());
viewHolder.distance.setText(Double.toString(event.getDistance()));
```

Assign the Values for the Current Position

```
ViewHolder viewHolder = (ViewHolder) convertView.getTag();
```

```
viewHolder.address.setText(event.getStreet());
viewHolder.rating.setRating(event.getRating());
viewHolder.distance.setText(Double.toString(event.getDistance()));
```

Lab 4.1 – Step 3
Optimize ListView with ViewHolder

Advanced Topics

- Multiple Row Types
- Modifying Data in Place
- Filtering
- Extending ListFragment or ListActivity
- Headers and Footers

Section 04
Android User Interface

Menus

Creating Menus

- Android offers a simple framework for you to add standard menus to your application.

- Menus can be configured through XML

Types of Menus

- Options
  - The primary collection of menu items for an activity, which appears when the user touches the MENU button. When your application is running on Android 3.0 or later, you can provide quick access to select menu items by placing them directly in the Action Bar, as "action items."
- Context
  - A floating list of menu items that appears when the user touches and holds a view that's registered to provide a context menu.
- Submenu
  - A floating list of menu items that appears when the user touches a menu item that contains a nested menu.

Example of Menu Configuration File

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/
android">
    <item android:id="@+id/new_game"
          android:icon="@drawable/ic_new_game"
          android:title="@string/new_game" />
    <item android:id="@+id/help"
          android:icon="@drawable/ic_help"
          android:title="@string/help" />
</menu>
```

Inflating a Menu

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.game_menu, menu);
    return true;
}
```

Responding to User Action

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle item selection
    switch (item.getItemId()) {
    case R.id.new_game:
        newGame();
        return true;
    case R.id.help:
        showHelp();
        return true;
    default:
        return super.onOptionsItemSelected(item);
    }
}
```

Lab 4.2
Creating a New Menu

Section 4
UI

Styles and Themes

Session Objectives

- Styles
- Themes

Section Review

1. What are Styles

2. What are Themes

3. UI Best Practices

What are Styles

- Similar to CSS on web pages
- Can specify properties such as orientation, height, padding, font color and background color
  - Define a style property
- Collection of properties that specify the look and format for a view
  - group style properties
- Defined in an XML resource that is separate from the XML that specifies the layout

Example of Applying style

```
  <TextView
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:textColor="#00FF00"
     android:typeface="monospace"
     android:text="@string/hello" />
<TextView
   style="@style/CodeFont"
   android:text="@string/hello" />

<?xml version="1.0" encoding="utf-8"?>
<resources>
   <style name="CodeFont"
parent="@android:style/
TextAppearance.Medium">
      <item
name="android:layout_width">fill_parent</
item>
```

Inheritance

```
<style name="GreenText" parent="@android:style/TextAppearance">
    <item name="android:textColor">#00FF00</item>
</style>
```

```
<style name="CodeFont.Red">
    <item name="android:textColor">#FF0000</item>
</style>
```

Attribute and Style Properties

- http://developer.android.com/reference/android/R.attr.html
- http://developer.android.com/reference/android/R.style.html

Lab 4.4
Using Styles

Section 05
Layouts

Layout Containers
GridView
Weight and Gravity
Layout Techniques and Best Practices

Session Objectives

- Using layouts

Section 05
Layout Containers

Layout Containers

- LinearLayout
- RelativeLayout
- GridLayout
- FrameLayout

LinearLayout

- Children are arranged in a line
- Can be horizontal or verticle
- Use weight to specify relative size of child elements
- Can be nested
- Don’t nest too deeply
  - Poor performance
  - Difficult to understand

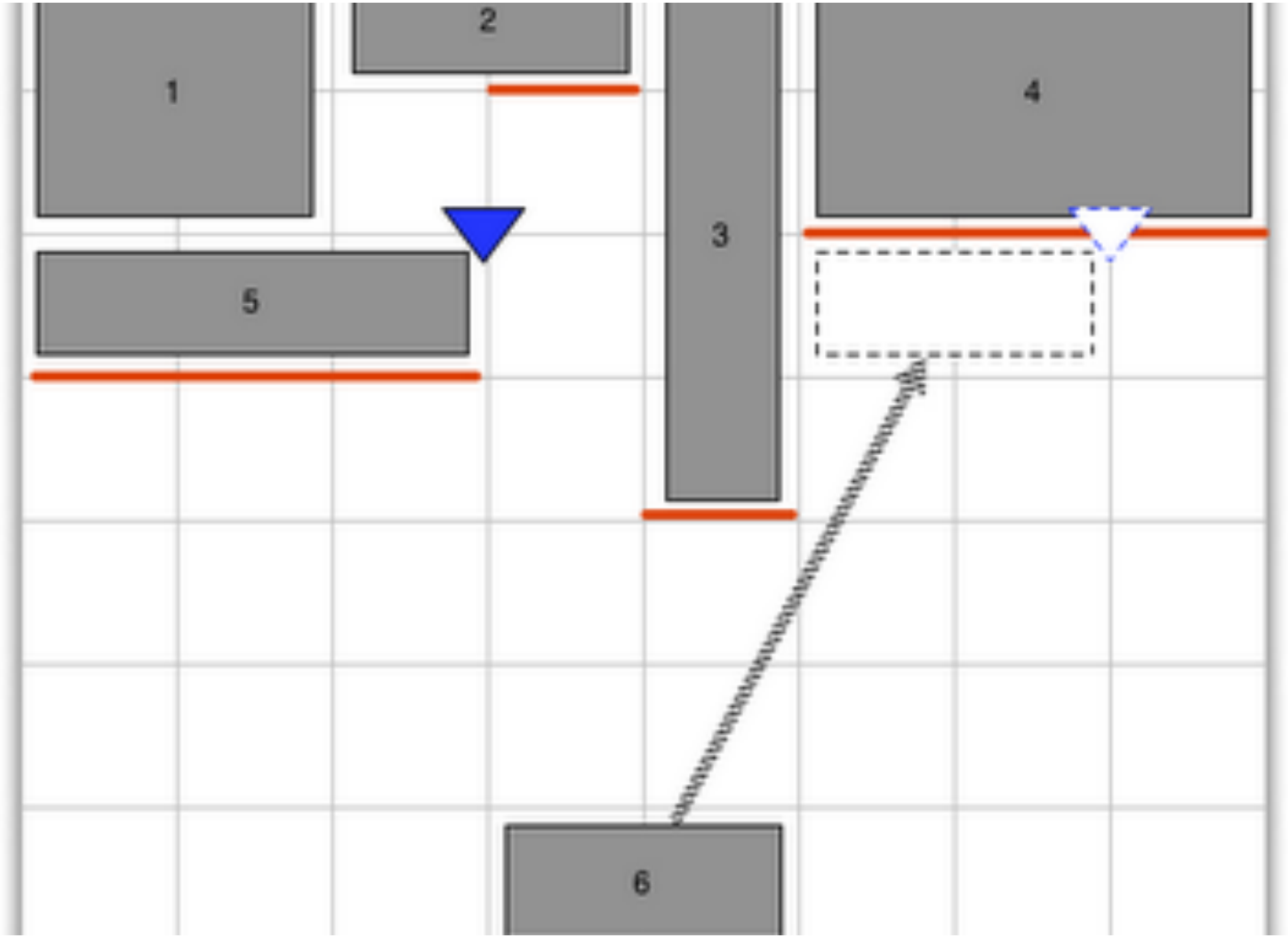
Relative Layout

- Should be “flat”
- Specify position of children “relative” to other children or the parent
- Better performance than LinearLayout

GridLayout

- Layout child elements in a flexible “infinite” grid using rows and columns
- Specify position of child elements into cells
- Use “layout_gravity” and “gravity” to align elements in cells

GridLayout Example



FrameLayout

- Children “overlap” each other

Section Review

1. Android provides many layout containers for different purposes.

Section 06
Activities

Activity Lifecycle
Creating Activities

Creating an Activity

- Create a subclass of Activity (or an existing subclass of it). In your subclass, you need to implement callback methods that the system calls when the activity transitions between various states of its lifecycle, such as when the activity is being created, stopped, resumed, or destroyed. The two most important callback methods are:

- onCreate()
  - You must implement this method. The system calls this when creating your activity. Within your implementation, you should initialize the essential components of your activity. Most importantly, this is where you must call setContentView() to define the layout for the activity's user interface.

- onPause()
  - The system calls this method as the first indication that the user is leaving your activity (though it does not always mean the activity is being destroyed). This is usually where you should commit any changes that should be persisted beyond the current user session (because the user might not come back).

Inflate the view

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.add_event);
}
```

Declare the Activity in the manifest

```
<manifest ... >
  <application ... >
      <activity android:name=".ExampleActivity" />
      ...
  </application ... >
  ...
</manifest >
```

Start an Activity

```
Intent intent = new Intent(this, SignInActivity.class);
startActivity(intent);
```

Intents

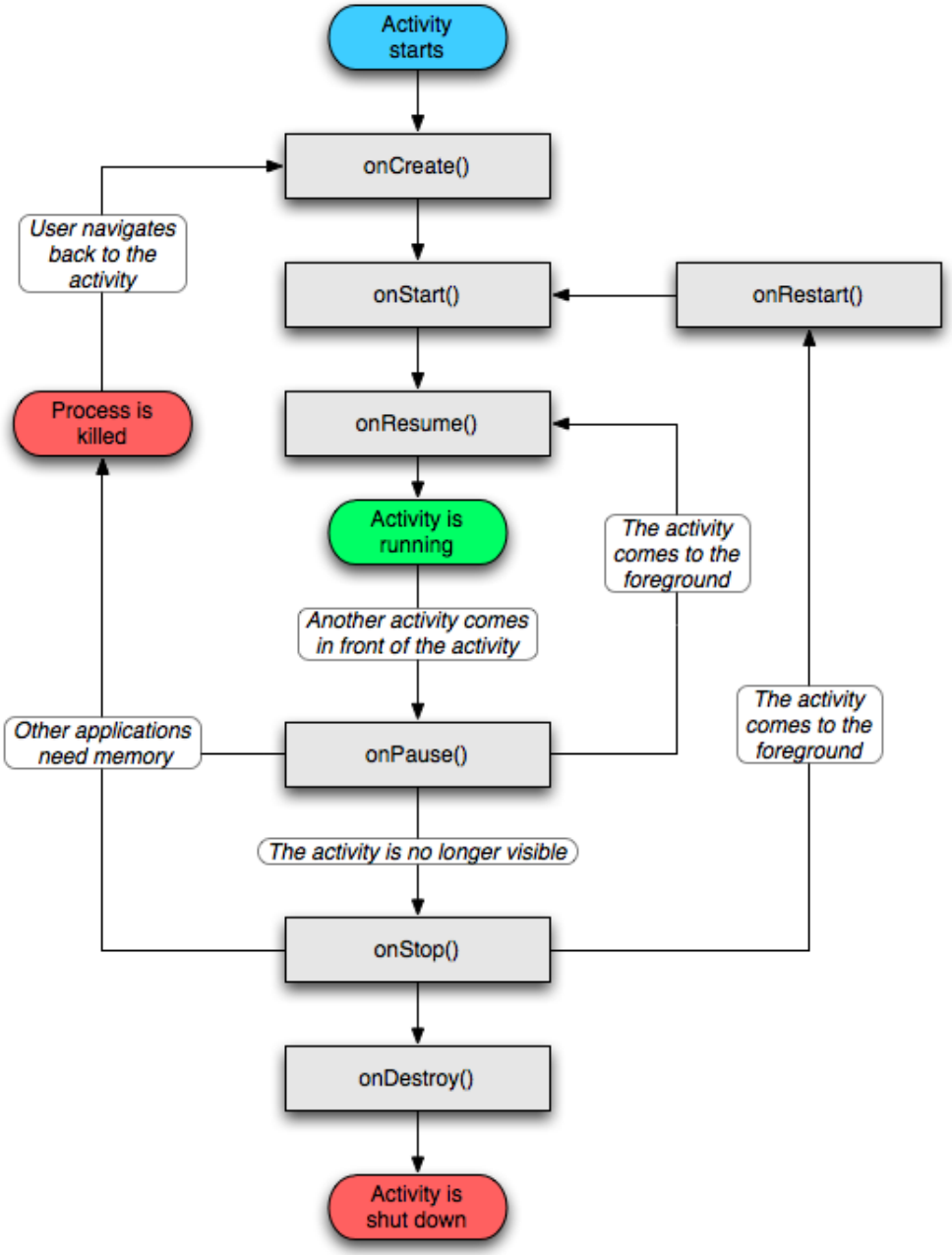
- Intents are Androids messaging system
- Intents are sent to the Android framework
- Android framework resolves intents and starts proper application component

- http://developer.android.com/guide/topics/intents/intents-filters.html

Shutting Down an Activity

- Shut down an activity by calling its `finish()` method

Activity Lifecycle



Summary of Activity Lifecycle Callback Methods

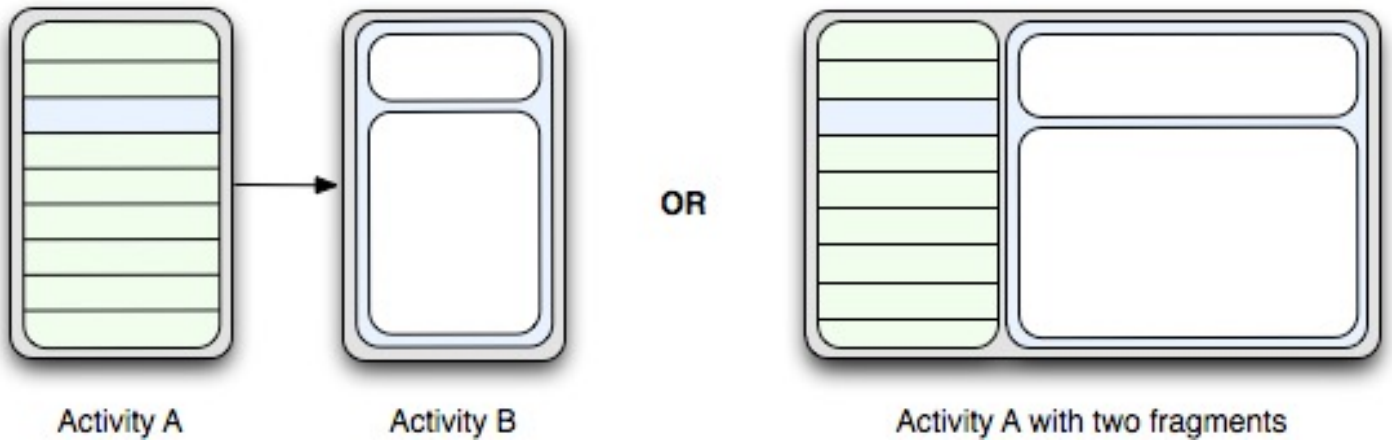
Method	Description	Killable after?	Next
onCreate()	Called when the activity is first created. This is where you should do all of your normal static set up — create views, bind data to lists, and so on. This method is passed a Bundle object containing the activity's previous state, if that state was captured (see Saving Activity State, later). Always followed by onStart().	No	onStart()
onRestart()	Called after the activity has been stopped, just prior to it being started again. Always followed by onStart()	No	onStart()
onStart()	Called just before the activity becomes visible to the user. Followed by onResume() if the activity comes to the foreground, or onStop() if it becomes hidden.	No	onResume() or onStop()
onResume()	Called just before the activity starts interacting with the user. At this point the activity is at the top of the activity stack, with user input going to it. Always followed by onPause().	No	onPause()
onPause()	Called when the system is about to start resuming another activity. This method is typically used to commit unsaved changes to persistent data, stop animations and other things that may be consuming CPU, and so on. It should do whatever it does very quickly, because the next activity will not be resumed until it returns. Followed either by onResume() if the activity returns back to the front, or by onStop() if it becomes invisible to the user.	**Yes**	onResume() or onStop()
onStop()	Called when the activity is no longer visible to the user. This may happen because it is being destroyed, or because another activity (either an existing one or a new one) has been resumed and is covering it. Followed either by onRestart() if the activity is coming back to interact with the user, or by onDestroy() if this activity is going away.	**Yes**	onRestart() or onDestroy()
onDestroy()	Called before the activity is destroyed. This is the final call that the activity will receive. It could be called either because the activity is finishing (someone called finish() on it), or because the system is temporarily destroying this instance of the activity to save space. You can distinguish between these two scenarios with the isFinishing() method.	**Yes**	*nothing*

Lab 6.1
Create a New Activity

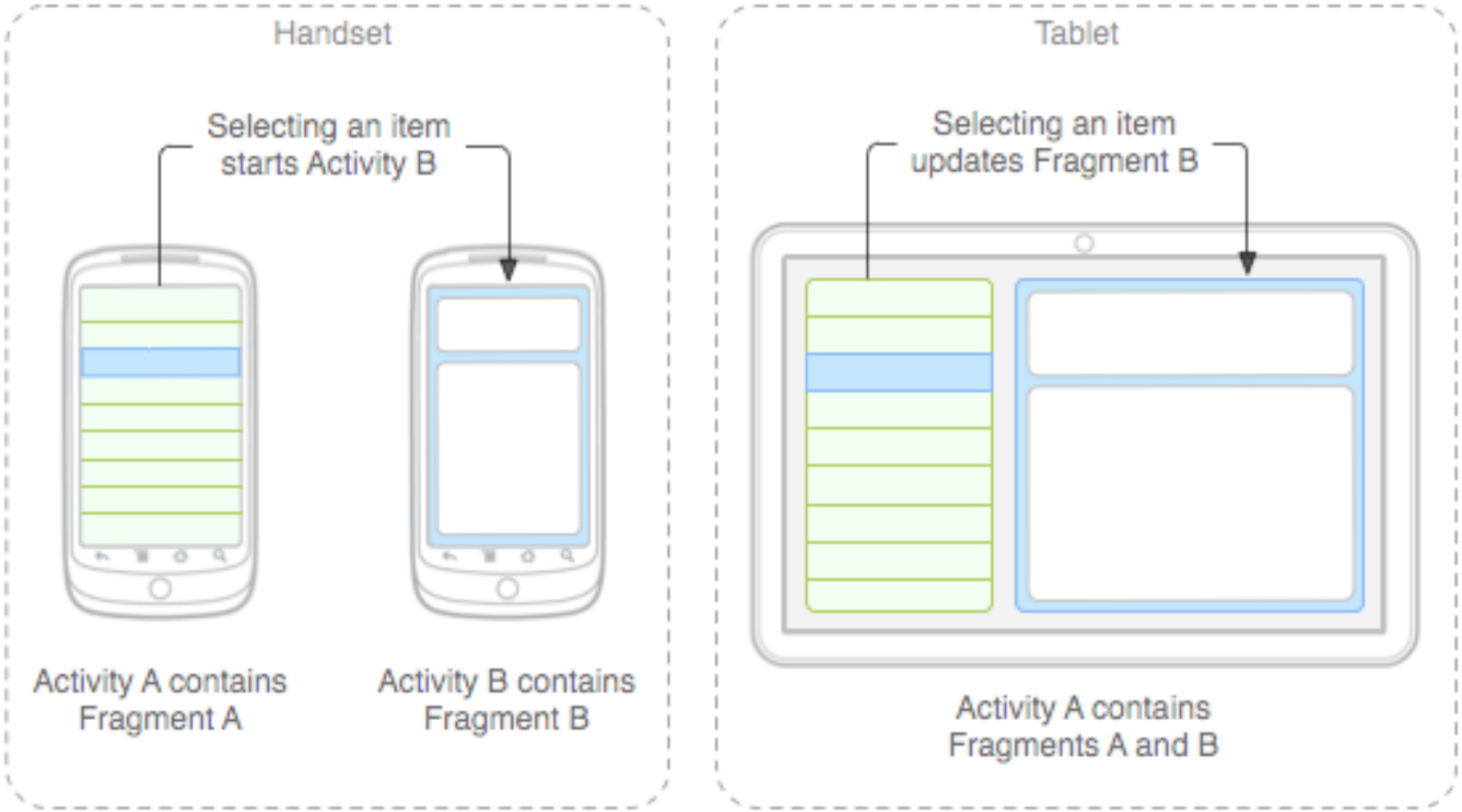
Section 07
Fragments

Fragments

- Fragments API allows re-use of interface components across activities
- Provides dynamic adjustment of UI in different devices without re-programming



Fragments on Handset and Table



Fragment Layouts

Here's res/layout/main.xml for handsets:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment class="com.example.android.TitlesFragment"
            android:id="@+id/list_frag"
            android:layout_width="match_parent"
            android:layout_height="match_parent"/>
</FrameLayout>
```

And res/layout-large/main.xml for tablets:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
  android:orientation="horizontal"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:id="@+id/frags">
  <fragment class="com.example.android.TitlesFragment"
```

How Fragments Work

How the application responds when a user selects an item from the list depends on whether Fragment B is available in the layout. If Fragment B is there, Activity A notifies Fragment B to update itself. If Fragment B is not in the layout, Activity A starts Activity B (which hosts Fragment B).

To implement this pattern for your application, it's important that you develop your fragments to be highly compartmentalized. Specifically, you should follow two general guidelines:

Do not manipulate one fragment directly from another.
Keep all code that concerns content in a fragment inside that fragment, rather than putting it in the host activity's code.
To avoid directly calling one fragment from another, declare a callback interface in each fragment class that it can use to deliver events to its host activity, which implements the callback interface. When the activity receives a callback due to an event (such as the user selecting a list item), it acts appropriately based on the current fragment configuration.

How Fragments work in Activity

For example, Activity A from above handles item selections like this:

```
/** This is a callback that the list fragment
(Fragment A) calls
    when a list item is selected */
public void onItemSelected(int position) {
  DisplayFragment fragB =
(DisplayFragment) getFragmentManager()
                          .findFragmentById(R.id.di
splay_frag);
  if (fragB == null) {
      // DisplayFragment (Fragment B) is not in
the layout,
      // start DisplayActivity (Activity B)
      // and pass it the info about the selected
item
      Intent intent = new Intent(this,
DisplayActivity.class);
      intent.putExtra("position", position);
```

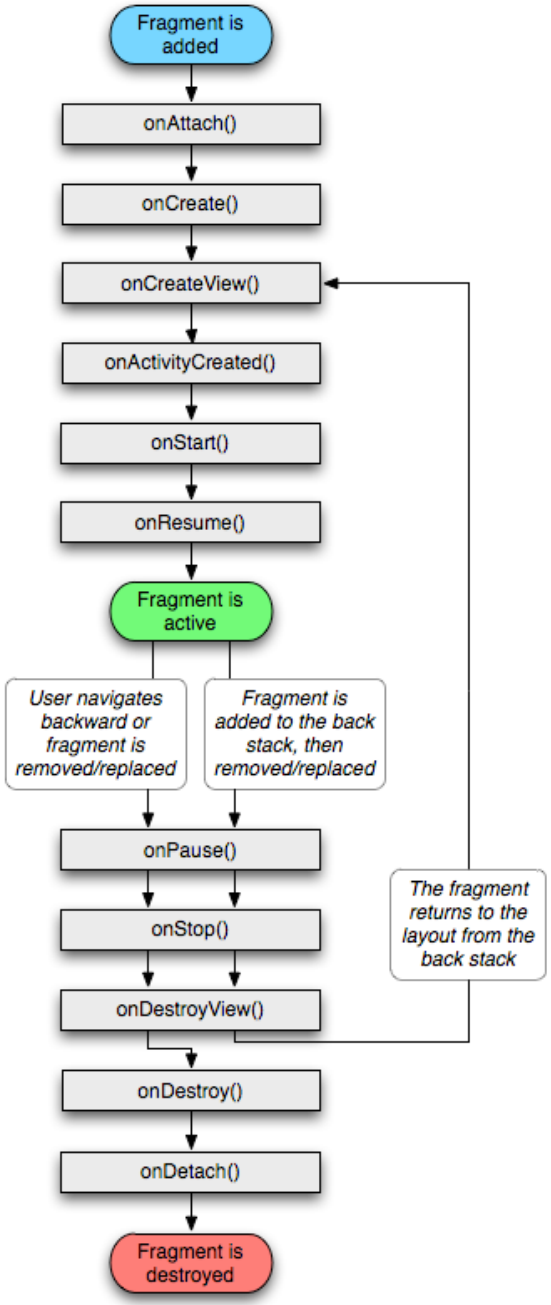
Fragments

- Introduced in Android 3.0
- Compatability library provides functionality for older versions of Android back to 1.6
- Packaged as Android Compatibility Package in "android-support-v4.jar"
- Available through SDK Updater

Definition

- Represents behavior of a portion of a user interface in an activity
- Multiple fragments can be combined into a single activity
- Used to build multi-pane Uis
- Has its own lifecycle
- Must always be embedded in an activity

Fragment Lifecycle



- Create fragments by subclassing `Fragment`
- Callback methods similar to `Activity`
- onCreate
  - Called when creating a fragment
- onCreateView
  - Called the first time the fragment draws its user interface
- onPause
  - Called when user is leaving fragment

Fragment Manager

- Fragment Manager
  - How many FMs are there
  - What does it do
  - How to access it
  - How to use it

Fragment Specializations

- DialogFragment
  - Displays a floating dialog
- ListFragment
  - Displays list of items managed by an adapter
- PreferenceFragment
  - Displays a hierarchy of `Preference` objects as a list
  - Similar to `PreferenceActivity`

Lab 7.1
Fragments API

Section 08
Storing and Retrieving Data

Internal Storage
Preferences
External Storage
Database Access

Session Objectives

- Internal Storage
- Preferences
- External Storage
- Databases

Section 08
Internal and External Storage

File System

- Android uses a Linux based file system that is accessible from an application
- Can be accessed using java.io as a FileOutputStream or FileInputStream
- Supports various permissions

int	MODE_APPEND	File creation mode: for use with `openFileOutput(String, int)`, if the file already exists then write data to the end of the existing file instead of erasing it.
int	MODE_MULTI_PROCESS	SharedPreference loading flag: when set, the file on disk will be checked for modification even if the shared preferences instance is already loaded in this process.
int	MODE_PRIVATE	File creation mode: the default mode, where the created file can only be accessed by the calling application (or all applications sharing the same user ID).
int	MODE_WORLD_READABLE	File creation mode: allow all other applications to have read access to the created file.
int	MODE_WORLD_WRITEABLE	File creation mode: allow all other applications to have write access to the created file.

File Access

- Writing Files
  - FileOutputStream
- Reading Files
  - FileInputStream

Files as raw resources

- Place files in "res/raw"

```
InputStream is = getResources().openRawResource(R.raw.resfile);
```

XML Resource Files

- XML files in "res/xml"
- Compiled by Android before being deployed to device
- Use a different API for access (not java.io)
  - Use XmlPullParser

Internal Storage

- On device
- Limited in size

External Storage

- On SD card when available
- Usually in "sdcard" directory
- Can be setup in emulator
- Uses java.io API for access
- You can acquire a File for this location by using the method `Environment.getExternalStorageDirectory()`

Lab 8.1

Write to Local Storage

Section 08
Shared Preferences

Shared Preferences

- Framework for saving and retrieving key/value pairs
- Uses primitive type data
  - Boolean
  - String
  - Float, int, long
- Persists across user sessions
- Can be shared or private
  - Shared with entire application

Preferences API

- getSharedPreferences
- getPreferences
- SharedPreferences.Editor
- put{data type}
- get{data type}

Example

```
public class Calc extends Activity {
    public static final String PREFS_NAME = "MyPrefsFile";

    @Override
    protected void onCreate(Bundle state){
       super.onCreate(state);

       // Restore preferences
       SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
       boolean silent = settings.getBoolean("silentMode", false);
       setSilent(silent);
    }

    @Override
    protected void onStop(){
       super.onStop();

      SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
      SharedPreferences.Editor editor = settings.edit();
      editor.putBoolean("silentMode", mSilentMode);

      // Commit the edits!
      editor.commit();
    }
}
```

Lab 8.2

Write to Preferences

Section 08
Database Access

RDBMS

- Android provides a local relation database called SQLite
  - Open source
  - Standard Compliance
  - Lightweigt
- Full Featured
  - Supports transactions
  - Functions
  - Foreign keys
  - triggers
- Uses Structured Query Language (SQL)

- Does not use jdbc, uses android.database

Creating the DBAdapter Helper Class

- A good practice for dealing with databases is to create a helper class to encapsulate all the complexities of accessing the database so that it's transparent to the calling code

- So, create a helper class called DBAdapter that creates, opens, closes, and uses a SQLite database.

SQLiteOpenHelper

- Abstract class
- Implements patterns for creating, opening and upgrading databases
- Wraps logic for deciding if a database needs to be upgraded before it is opened
- getWritableDatabase
- Fallback to getReadableDatabase

Upgrading the Database

- To upgrade the database, change the DATABASE_VERSION constant in the DBAdapter class to a value higher than the previous one.

Lab 8.3

Write to Database

Section Review

1. What are the storage alternatives in Android

2. Which storage alternative is appropriate for each kind of use

Section 9
Content Providers

Section Objectives

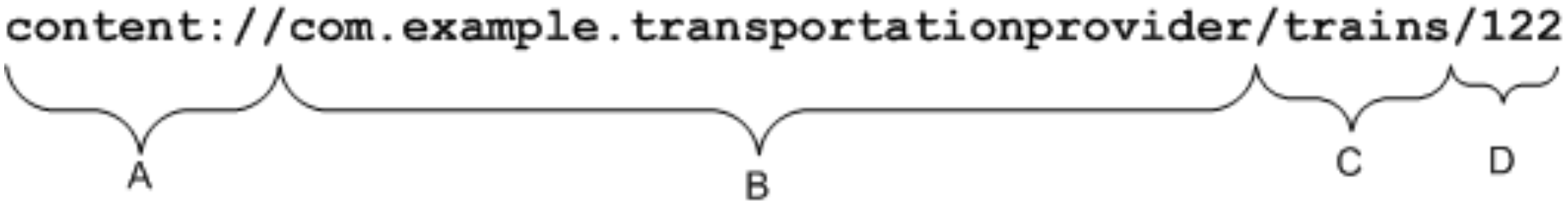
- Understand Content Providers

What is a ContentProvider

- The ContentProvider implements a standard set of methods to permit an application to access a data store
- Can use any form of data storage mechanism available on the Android platform, including files, SQLite databases
- A ContentProvider's data is accessed by an Android application through a Content URI
- Query is done using SQL
- Iterate through CP using a Cursor

What is a ContentProvider

- A ContentProvider's data is accessed by an Android application through a Content URI



- A) Standard prefix for content providers
- B) URI authority for content provider
- C) Kind of data, multiple segments allowed
- D) ID of record being requested

When to use a ContentProvider

- CPs are more complicated than direct local database access
- Use a CP only if you need to share data with another application or provide access to your data with search
- Otherwise just use a local database

Lab 9.1

Create Content Provider

Section Review

1. What is a content provider?

Section 10
Asynchronous Tasks

Thread
AsyncTasks

Section Objectives

- Understand techniques for removing processing from the main UI thread

AsyncTask

- AsyncTask enables proper and easy use of the UI thread. This class allows to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.

- An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread. An asynchronous task is defined by 3 generic types, called Params, Progress and Result, and 4 steps, called onPreExecute, doInBackground, onProgressUpdate and onPostExecute.

Declaring an AsyncTask

```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
    protected Long doInBackground(URL... urls) {
        int count = urls.length;
        long totalSize = 0;
        for (int i = 0; i < count; i++) {
            totalSize += Downloader.downloadFile(urls[i]);
            publishProgress((int) ((i / (float) count) * 100));
        }
        return totalSize;
    }
    protected void onProgressUpdate(Integer... progress) {
        setProgressPercent(progress[0]);
    }
    protected void onPostExecute(Long result) {
        showDialog("Downloaded " + result + " bytes");
    }
}
```

AsyncTask's generic types

- The three types used by an asynchronous task are the following:

1. Params, the type of the parameters sent to the task upon execution.
2. Progress, the type of the progress units published during the background computation.
3. Result, the type of the result of the background computation.

- Not all types are always used by an asynchronous task. To mark a type as unused, simply use the type Void.

Executing an AsyncTask

```
new DownloadFilesTask().execute(url1, url2, url3); }
```

4 Steps when asynchronous task is executed

1. onPreExecute(), invoked on the UI thread immediately after the task is executed. This step is normally used to setup the task, for instance by showing a progress bar in the user interface.
2. doInBackground(Params...), invoked on the background thread immediately after onPreExecute() finishes executing. This step is used to perform background computation that can take a long time. The parameters of the asynchronous task are passed to this step. The result of the computation must be returned by this step and will be passed back to the last step. This step can also use publishProgress(Progress...) to publish one or more units of progress. These values are published on the UI thread, in the onProgressUpdate(Progress...) step.
3. onProgressUpdate(Progress...), invoked on the UI thread after a call to publishProgress(Progress...). The timing of the execution is undefined. This method is used to display any form of progress in the user interface while the background computation is still executing. For instance, it can be used to animate a progress bar or show logs in a text field.
4. onPostExecute(Result), invoked on the UI thread after the background computation finishes. The result of the background computation is passed to this step as a parameter.

Cancelling a task

- A task can be cancelled at any time by invoking cancel(boolean). Invoking this method will cause subsequent calls to isCancelled() to return true. After invoking this method, onCancelled(Object), instead of onPostExecute(Object) will be invoked after doInBackground(Object[]) returns. To ensure that a task is cancelled as quickly as possible, you should always check the return value of isCancelled() periodically from doInBackground(Object[]), if possible (inside a loop for instance.)

Threading Rules

- There are a few threading rules that must be followed for this class to work properly:
  - The task instance must be created on the UI thread.
  - execute(Params...) must be invoked on the UI thread.
  - Do not call onPreExecute(), onPostExecute(Result), doInBackground(Params...), onProgressUpdate(Progress...) manually.
  - The task can be executed only once (an exception will be thrown if a second execution is attempted.)

Lab 10.1
Creating an AsyncTask

Section Review

1. Why are asynchronous tasks necessary?

2. How are asynchronous tasks created?

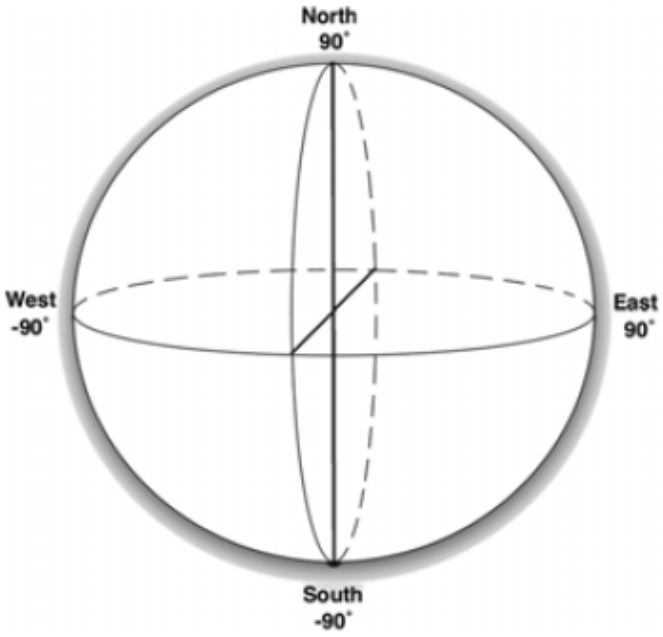
Section 11
Maps and Location

Google Maps
Location

Section Objectives

- Use the Google Maps API for Android
- Learn how to work with the Location services in Android

Review of Latitude and Longitude



- (0,0) is in the Atlantic Ocean, about 380 miles (611 kilometers) south of Ghana and 670 miles (1078 km) west of Gabon.
- Chicago is at negative -88 longitude and positve 42 latitude (roughly!)
- The order of latitude and longitude is important
- Some Google Maps APIs also uses micro latitude and longitude which requires multiplication by 1E6. This will usually be obvious because your location will be in the Atlantic Ocean!

Mapping Classes

Map Classes

MapView

- A View which displays a map (with data obtained from the Google Maps service). When focused, it will capture keypresses and touch gestures to pan and zoom the map. It can also be controlled programmatically (getController()) and can draw a number of Overlays on top of the map (getOverlays()).

- The map can be displayed in a number of modes; see setSatellite(boolean), setTraffic(boolean), and setStreetView(boolean).). It also draws a Google logo in the bottom-left corner.

- The preferred zoom mechanism is the built-in zoom, see setBuiltInZoomControls(boolean). As the user pans the map, zoom controls will automatically be shown at the bottom of the MapView.

GoogleMap

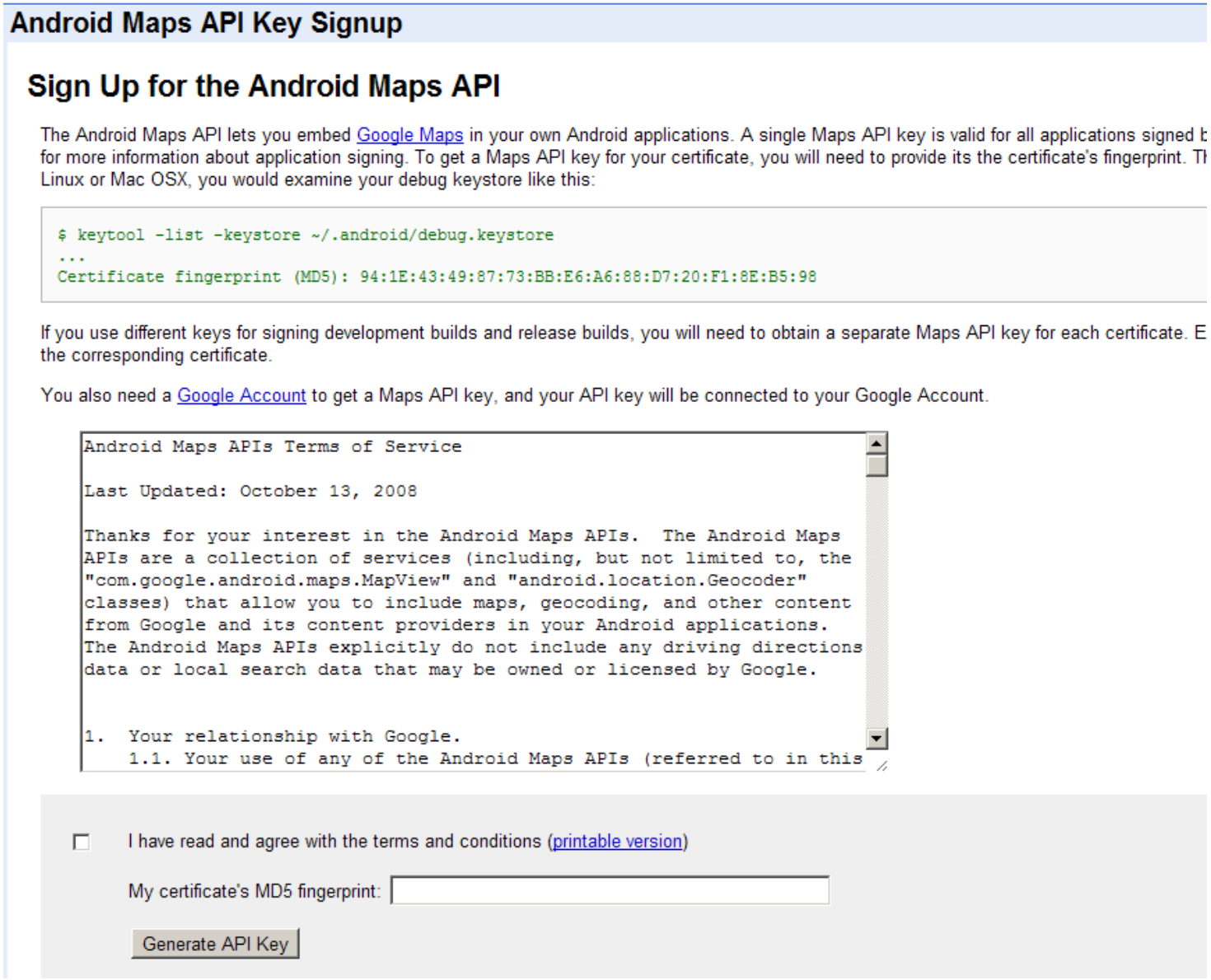
- A MapView can only be constructed (or inflated) by a MapActivity. This is because it depends on threads which access the network and filesystem in the background; these threads must be shepherded by the lifecycle management in MapActivity.

- Map tiles are cached on the filesystem in your application's directory. The cache is auto-managed so you don't need to do anything with it, and can delete it at any time.

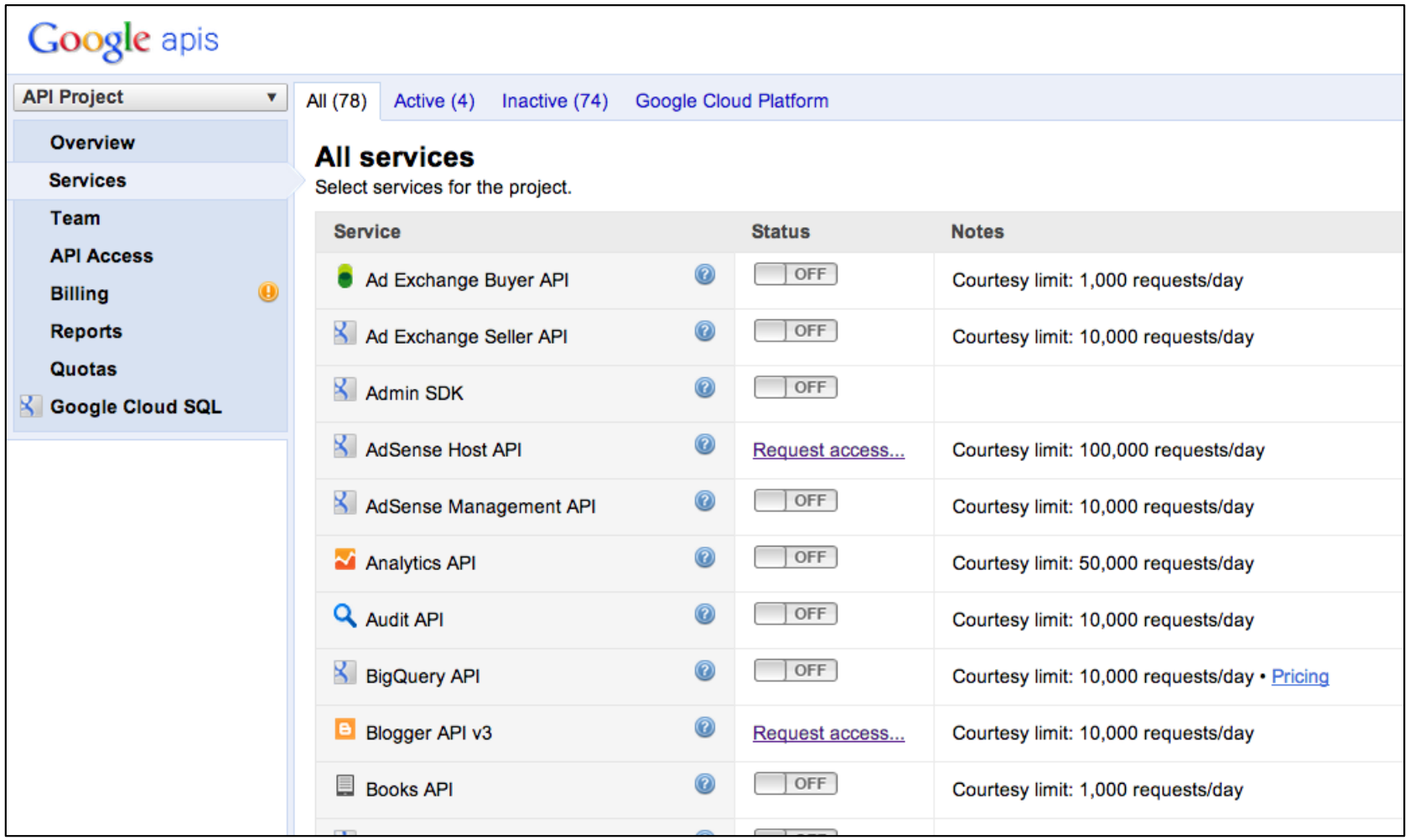
Maps API Key

- In order to display Google Maps data in a MapView, you must register with the Google Maps service and obtain a Maps API Key.
- For information about how to get a Maps API Key, see Obtaining a Maps API Key.
- Once you have a Maps API Key, you need to reference it in the application manifest

API Signup



Enable Google Maps v2 Service



Displaying a map

- Declare a view in layout (GoogleMap)
- Create an activity
- Inflate view
- Create camera (CameraUpdate)
- Set center of map (LatLng)
- Set zoom level (CameraUpdate)
- Display map (animate)

Classes

CameraUpdate	Defines a camera move.
CameraUpdateFactory	A class containing methods for creating `CameraUpdate` objects that change a map's camera.
GoogleMap	This is the main class of the Google Maps Android API and is the entry point for all methods related to the map.
GoogleMapOptions	Defines configuration GoogleMapOptions for a `GoogleMap`.
MapFragment	A Map component in an app.
MapsInitializer	Use this class to initialize the Google Maps Android API if features need to be used before obtaining a map.
MapView	A View which displays a map (with data obtained from the Google Maps service).
Projection	A projection is used to translate between on screen location and geographic coordinates on the surface of the Earth (`LatLng`).
SupportMapFragment	A Map component in an app.
UiSettings	Settings for the user interface of a GoogleMap.

Lab 11.1
Displaying A Map

Display Items on a Map using Marker

- Create Marker objects
- Configure Marker objects
  - Icon
  - Popup Text
  - Position
- Add Marker to the map

Lab 11.2
Markers on a Map

Location Provider Technology

Accuracy	Power Usage	Technology
20ft	**High**	**Autonomous GPS, Provider: gps** 1. uses GPS chip on the device 2. line of sight to the satellites 3. need about 7 to get a fix 4. takes a long time to get a fix 5. doesn't work around tall buildings
200ft	**Medium – Low**	**Assisted GPS (AGPS), Provider: network** 1. uses GPS chip on device, as well as assistance from the network (cellular network) to provide a fast initial fix 2. very low power consumption 3. very accurate 4. works without any line of sight to the sky 5. depends on carrier and phone supporting this (even if phone supports it, and network does not then this does not work)
5300ft / 1mile	**Low**	**CellID lookup/WiFi MACID lookup, Provider: network or passive** 1. very fast lock, and does not require GPS chip on device to be active 2. requires no extra power at all 3. has very low accuracy; sometimes can have better accuracy in populated and well mapped areas that have a lot of WiFi access points, and people who share their location with Google

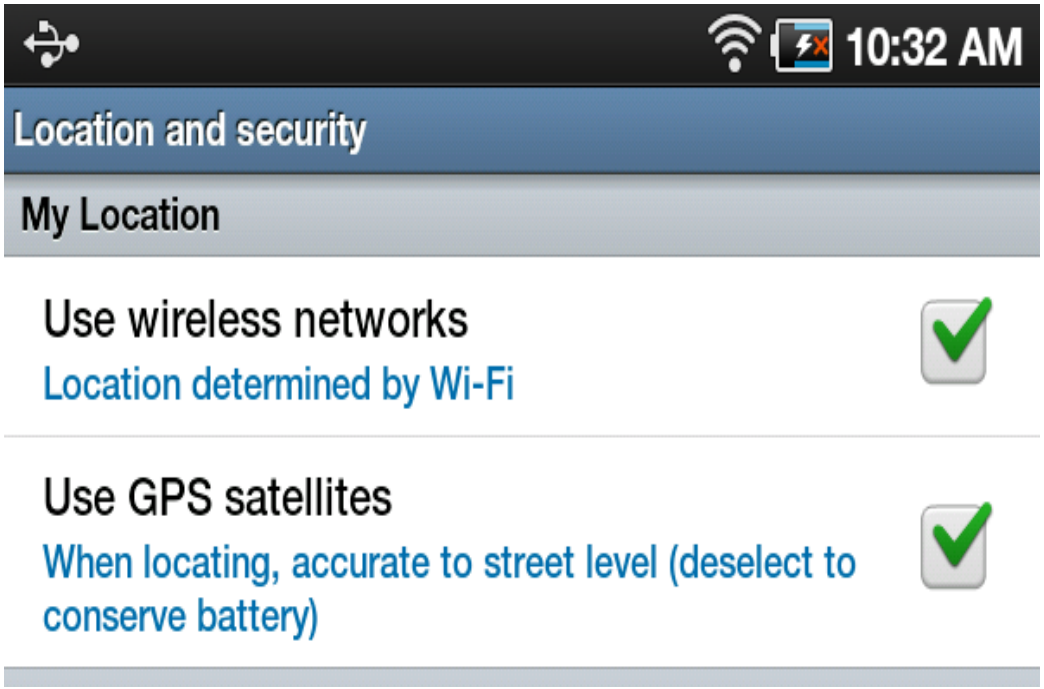
Network Providers

- There are 3 network providers in Android. They are:

- **gps –> (GPS, AGPS):** Name of the GPS location provider. This provider determines location using satellites. Depending on conditions, this provider may take a while to return a location fix. Requires the permission android.permission.ACCESS_FINE_LOCATION.
- **network –> (AGPS, CellID, WiFi MACID):** Name of the network location provider. This provider determines location based on availability of cell tower and WiFi access points. Results are retrieved by means of a network lookup. Requires either of the permissions android.permission.ACCESS_COARSE_LOCATION or android.permission.ACCESS_FINE_LOCATION.
- **passive –> (CellID, WiFi MACID):** A special location provider for receiving locations without actually initiating a location fix. This provider can be used to passively receive location updates when other applications or services request them without actually requesting the locations yourself. This provider will return locations generated by other providers. Requires the permission android.permission.ACCESS_FINE_LOCATION, although if the GPS is not enabled this provider might only return coarse fixes.

Location Settings on Device

- When only **"use wireless networks"** is checked, then CellID/MACID lookups are used first, and the "network" provider uses this, and gets about 200ft-1mile accuracy.
- When only **"use GPS satellites"** is checked, then, GPS is used, and the "gps' provider uses this, and gets less than 10ft accuracy.



Gotchas

- When attaching a location listener to a location provider (regardless of the exact kind of provider), it is important to execute this listener code in the main thread of your activity or service. If you run this in any background thread (like using an executor) then this will not work. The listener code itself has to be on the main thread. You can kick of a task in an executor in the listener, but the listener itself must run in the main thread of the app.
- Some devices will require you to detach the listener and then reattach it periodically. This is not a bad idea, since it eliminates the possibility of the listener going stale.
- You can switch providers at any time, and switch out a fine provider with a coarse one to save battery life, for example. There is really no limit to how often you can switch out providers. You can also do this when you discover that a provider is unavailable, via a call to the registered location listener.
- Finally, the parameters you pass to the location provider, when you attach your location listener are totally optional. The device you are running on will decide whether to respect those parameters or not. So it's best never to rely on them. Also, different devices behave differently with reporting location movements. Eg, the Droid X and Droid 2 are very zippy about providing location updates, they will do it every second and provide you with a very high level of accuracy. The Samsung Galaxy S is less zippy with the location updates. The HTC Incredible is better than the Galaxy S, but not as zippy with the frequent updates as the Droid X and Droid 2. Finally, the LG Ally is one of the worst with location updates, providing very infrequent updates, and very poor accuracy (even on autonomous GPS). So don't make any assumptions about the accuracy and frequency of updates – you have to test it on real devices on real networks in real life situations.

Lab 11.3
Acquiring Location

Section 12
RESTful Services
HTTP, XML and JSON

Making HTTP Requests
XML Parsing
JSON

Section Objectives

- Understand techniques for making HTTP requests in the Android SDK
- Understand the various options for parsing XML within the Android SDK
- Review specific XML parsing options
  - DOM
  - SAX
  - SAX Helper classes in Android
  - SAX Pull Parser
- Select an appropriate Http request mechanism and an XML parsing approach and implement them in a lab

HTTP and Parsing XML
Making HTTP Request

Making HTTP Requests
XML Parsing
Code Review

Making HTTP Requests in the Android SDK

- Using java.net
  - Included in standard JDK
- Using org.apache.http
  - Simpler API
  - No checked exceptions (except for input stream)
  - Not included in JDK
  - Available in android.jar
  - Harder to unit test outside of Android
    - Requires downloading of jar

AndroidHttpClient

- New to Android 2.2
- Package android.net.http
- Implement HttpClient
- Provides
  - SSL Management
  - Useful utility methods
- No automatic cookie storage

HTTP and Parsing XML
XML Parsing

Making HTTP Requests
XML Parsing
Code Review

Processing XML

- DOM
- SAX
- Android SAX
- SAX Pull Parser

DOM Parsing

- DOM object remains resident
- Can search document multiple times
- Takes longer to build a DOM then to single pass an input stream
- Powerful search techniques such as XPATH
- Most DOM parsers use a SAX parser internally to build the DOM

XML Pull Parsing

- XML Pull Parsing refers to the process of parsing XML as a stream rather than building a tree (DOM) or pushing events out to client code (SAX).
- The main goal of XML Pull Parsers is to optimize tasks where all elements in a document must be parsed and processed.
- This approach is called pull parsing because the parser only parses what is asked for by the application rather than passing all events up to the client application.
- The pull approach of this parsing model results in a very small memory footprint (no document state maintenance required), and very fast processing (fewer unnecessary event callbacks).

SAX

- Code in package java .xml
- Single Pass
- Must provide event handler code
- XML Processing Events
  - Starting a new tag
  - Ending tag
- SAX issues
  - Can’t pass over XML again
- Android Issues
  - SDK provides utility class for processing XML
  - android.util.Xml
    - parse(String xml, ContentHandler contentHandler)
  - What are the advantages of using android.util.Xml

Android XML Pull Parsing

- Special Android classes to support pull parsing
- See android.util.Xml

Using the SAX Parser

- Special helper class to combine raw helper classes
- Coding simpler

SAX Error Handling

- Android has special class
- DefaultHandler

HTTP and Parsing XML
Code Review

Making HTTP Requests
XML Parsing
Code Review

Garage Sale Domain Object

Title	Great garage sale this weekend in Verona
Description	Great stuff for sale. Significantly downsized two years ago when we moved to Wisconsin and finally have the time to have a sale. We want to PURGE!! Furniture, clothes and more.
Address City / State / Zip	410 Dunhill Drive Verona, WI 53593
Latitude and Longitude	43.005740 -89.538853
Rating	3.5

Code Review

- Event Object
- XML file to parse
- HTTP Request Techniques
- XML Parsing Techniques
- Unit Testing

Other Alternatives

- JSON
  - Faster to send
  - Faster to parse
  - More natural in the Android API
- Google Protocol Buffers
- jTidy for parsing HTML
  - Adding extra libraries to Android

Lab 12.1
Parsing Android Compiled XML

Making Network Requests

- << insert slide showing radio usage during network access>>

Improved Network Requests

- << insert slide showing radio usage during network access>>

Comparison Of Battery Usage

- << insert slide showing radio usage during network access>>

Lab 12.2
Creating a JSON Service

Section Review

1. What alternatives are there in the Android SDK for making HTTP Requests. Compare them and describe the pros and cons of each.

2. What alternatives exist for parsing XML. Compare them and describe the pros and cons of each.

Section 13
Application Fundamentals Review

Fundamental Components

Session Objectives

- Activities
- Intents
- Services
- Broadcast Receivers
- Content Providers
- Application Manifest
- Additional Components
  - Notification Manager
  - Alarm Manager

Activities

- Single screen with a user interface
- Independent but work together
- Can be invoked from other applications

- Extends the Activity class

Service

- Perform long-running operations in the background
- No User Interface
- Can bind to other services or activities

- Extends the Service class

Content Provider

- Manages shared application data
- Provides consistent interfaces to data
  - Restful
  - CRUD operations
- Data Store backing options
  - SQLite DB
  - File System
  - Web
- Can consume data from other Content Providers

- Extends the `ContentProvider` class

Broadcast Reciever

- Respond to system wide messages
- Messages can be initiated by system or another app
- 2 kinds
  - Delivered asynchronously to all receivers
  - Delivered in priority order Should be light weight, work should be passed to services
- Usually Short-Lived

- Extends the BroadcastReceiver class

Intents

- Messages that link app components together
  - Explicit – launch a specific component
  - Implicit – launch any component which is registered to handle intent

- Describes an operation to be performed
  - Requested action
  - Data used by the action
  - Extra metadata

Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>

<manifest>
    <uses-permission />

    <application>
        <activity> … </activity>
        <service> … </service>
        <provider> … </provider>
        <receiver> … </receiver>
    </application>

</manifest>
```

Notification Manager

- Record events without interrupting user
- Appears at the top of the page
- Can provide rich notifications
  - Link to activity
  - Act as mini-activity

Alarm Manager

- Schedule future tasks
- Rich scheduling techniques
- Avoid long running services

Lab 13.1 – Build Sync Service

- Create a service to upload data to cloud
- Create a service to download data from cloud
- Services should create notifications when new data is found
- Create broadcast receiver to be triggered by alarm manager
- Register broadcast receiver with alarm manager

Lab 13.1

Build Sync Service

Section 14
WebView

Session Objectives

- Overview
- Web View
- Debugging Web Apps with JavaScript

WebView

- Browser-based apps or Native apps
- Even native apps may sometimes need to expose a web page
- WebView is native app view element for displaying web page
  - Able to display a single web page
  - Able to render CSS
  - Able to execute JavaScript

WebView Limitations

- WebView does not include
  - Navigation controls
    - Back / Forward
  - Address bar
    - URL must be provided by application

Using WebView

- Add WebView to layout using using <WebView> element
- WebView is a View
- Load page using “loadUrl” method
- Don’t forget to add internet permissions to manifest

```
<uses-permission android:name=“android.permission.INTERNET />
```

Calling JavaScript

- App can run JavaScript functions on web page
- Web page can call functions (methods) in native app

Enabling JavaScript

- To enable JavaScript binding between page and app

```
WebSettings webSettings = webView.getSettings();
webSettings.setJavaScriptEnabled(true);
```

Create a JavaScriptInterface class

- Build a new JavaScriptInterface class
- Define methods that can be run by the page
- Register the JavaScriptInterface class with the Web View
- Modify the page to call the JSI method

Problems working with JavaScript

- Timing
- JavaScript failures
- Only works with Internet connection

Section 15
Best Practices

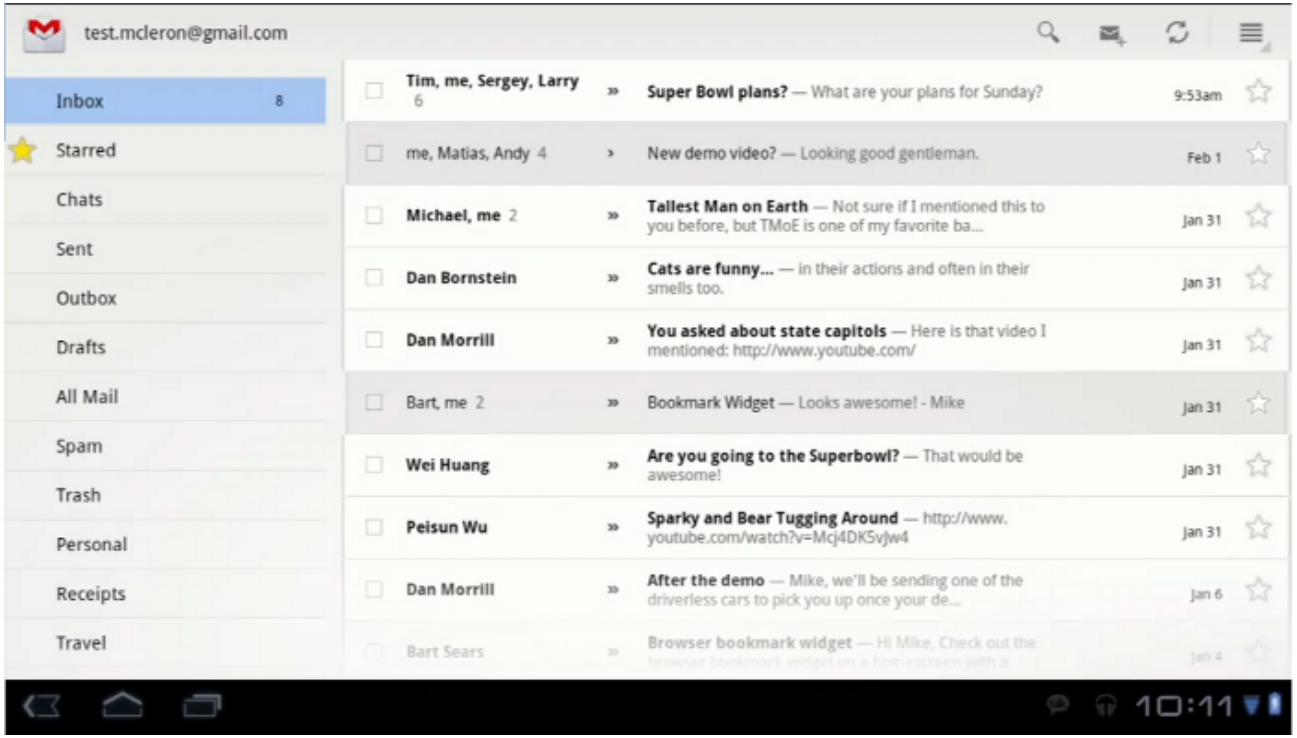
Development Best Practices
Tablet Best Practices

Make Full Use of the Screen

- Don't just scale up phone screen
- Screen will have too much blank space

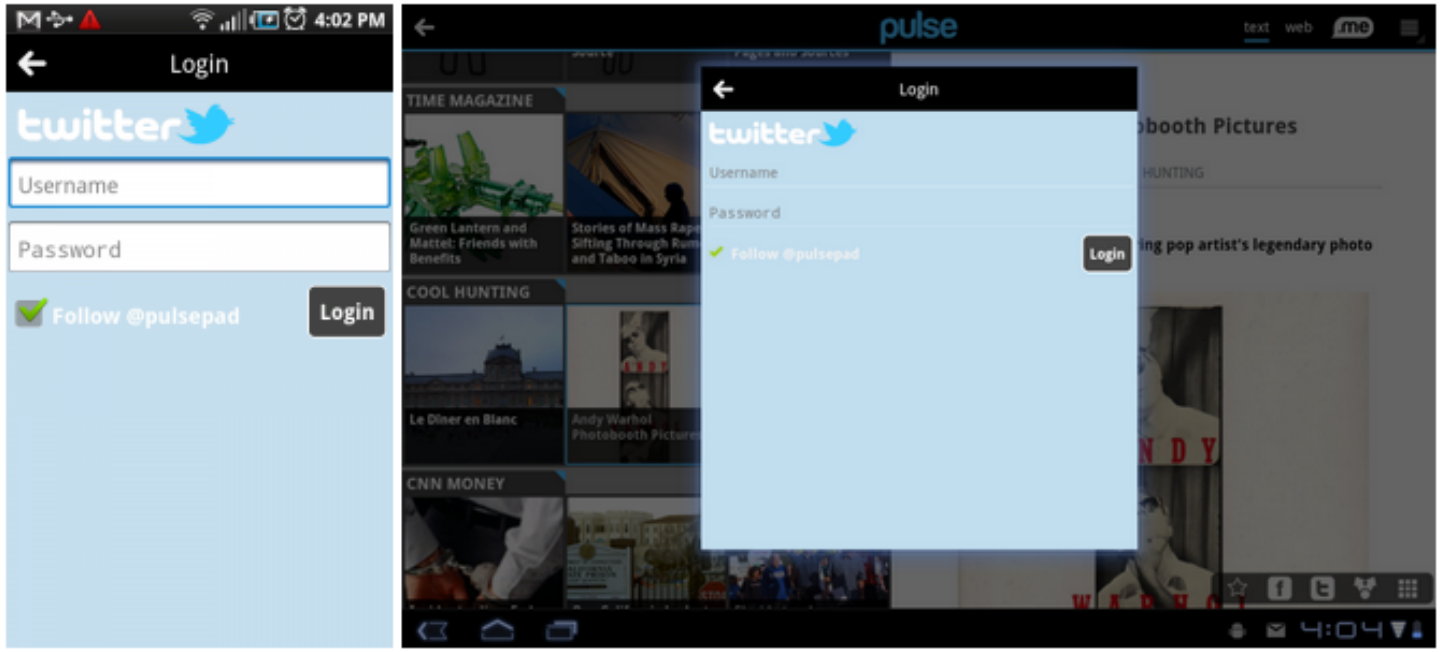
Make Full Use of the Screen

- Use Fragments to utilize extra space



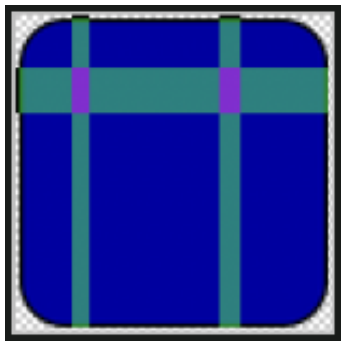
Use Dialogs

- Dialogs can show smaller components



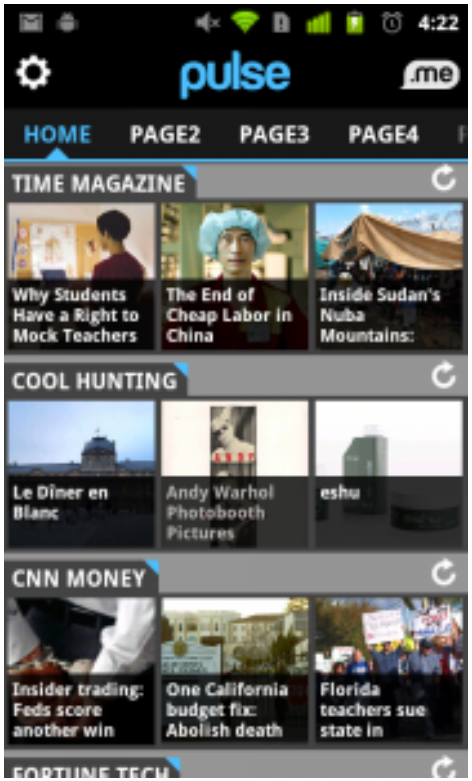
Use 9 patch

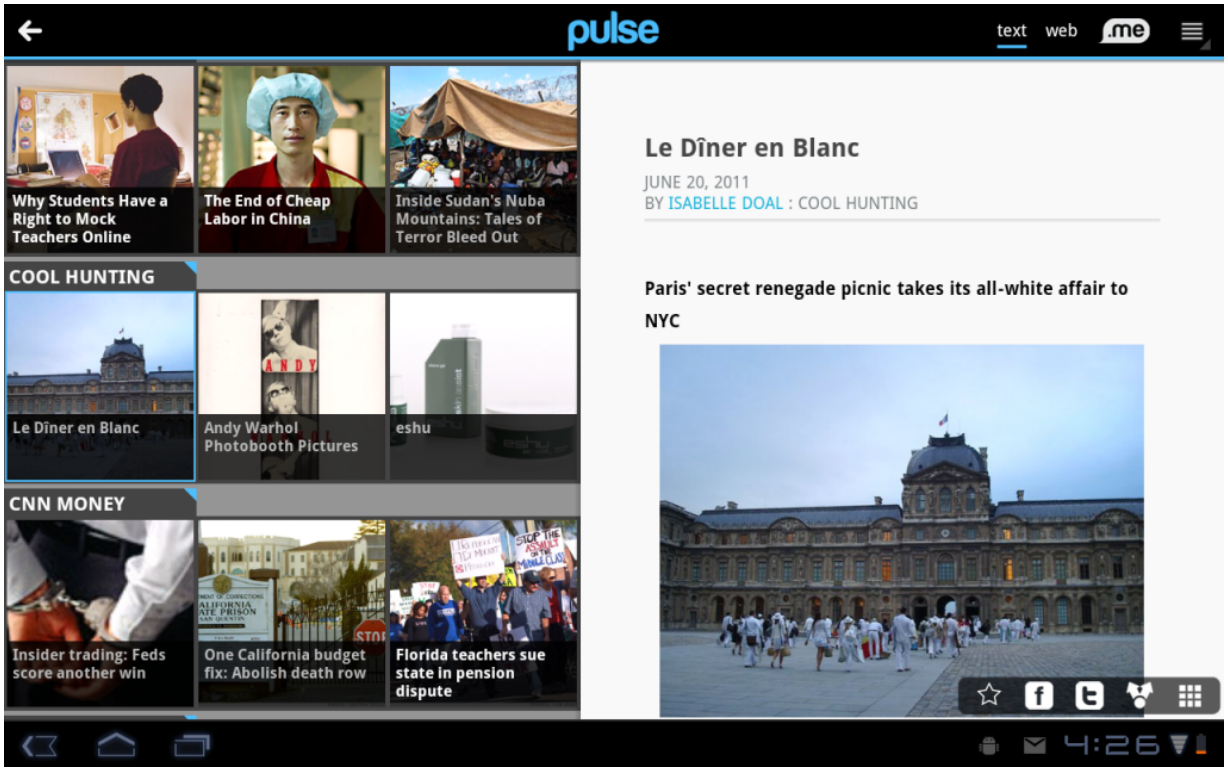
- Use 9 patch images to stretch images



Design for both Landscape and Portrait Modes

- Use different layouts for each orientation
- Users have strong preferences for their favored orientation



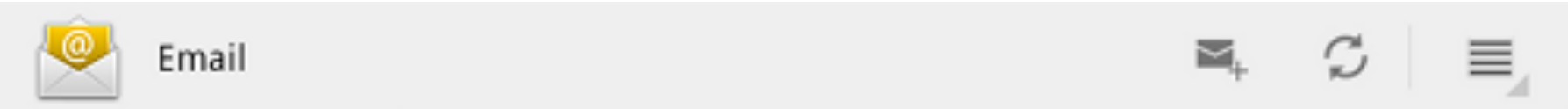


Create images using XML and drawable

- Images such as buttons and backgrounds often need to be created by designers
  - Problems updating
- Android allows the creation of complex images by specifying them as drawables using XML
  - Gradients, Porter/Duff
- This allows the images to be manipulated and re-used by the developer without designer input

Honeycomb UI Conventions

- No menu button
- Action Bar
  - Top left used for going to front of application



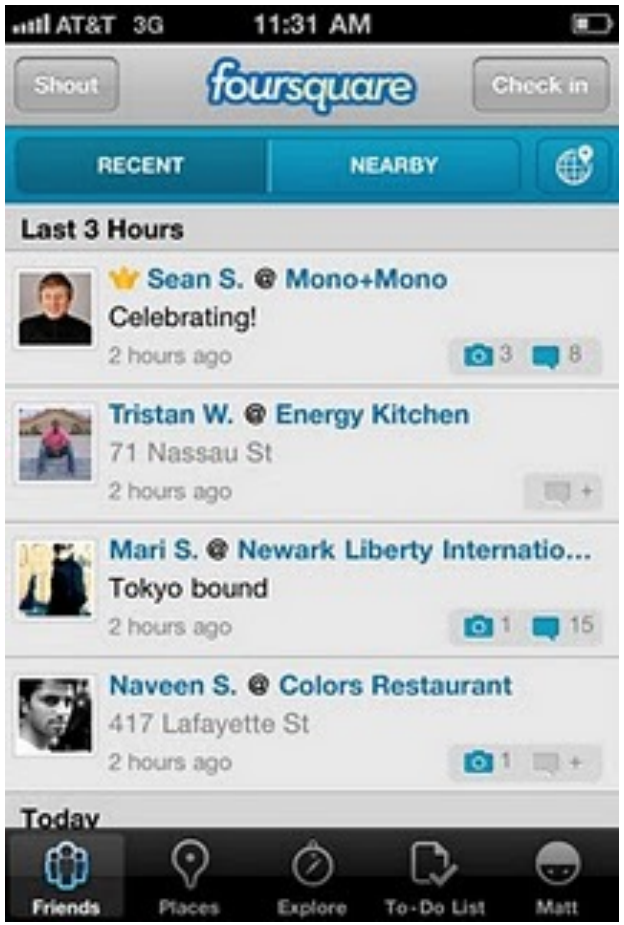
Design Pattern for Refreshing Data

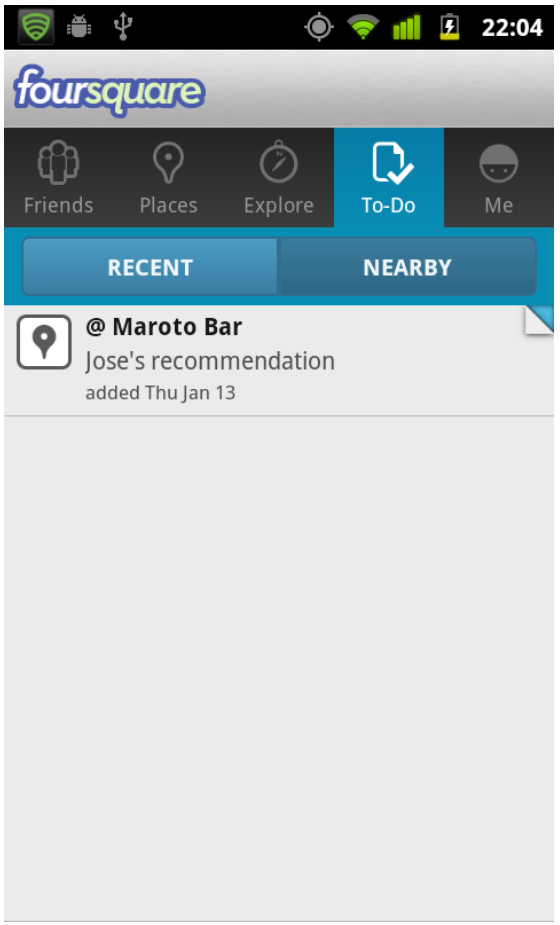
- Server Update and Notification
  - Alarm
  - Broadcast Receiver
  - Service
  - Notification
- Alarm, create an alarm when the application starts to schedule a server update
- Broadcast Reciever, alarm starts broadcast receiver which starts service to perform update
- Service creates a notification when complete
- Notification takes user to display of events with new items highlighted in red (or just new all new items or new items on top)

Using an iOS Design for an Android App

- Use top tabs
- No back button
- Use Android icons
- Use ActionBar

Use Top Tabs





Use ActionBar

Source: Evernote