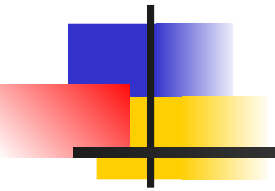
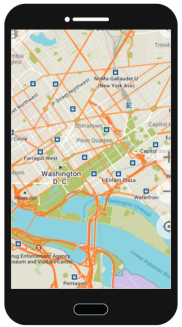


CS476/576: Programming Models for Emerging Platforms

Android Programming

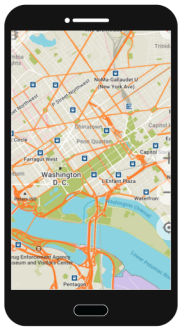
What role does mobile phone play?



Apps

Most people's
primary computing device

What role does mobile phone play?



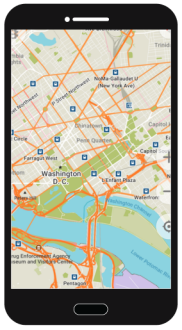
Apps

Most people's primary computing device

Snapchat

What functionality is on the device?

What role does mobile phone play?



Apps

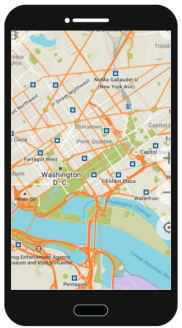
Most people's primary computing device

1. Camera
2. Location Services
3. User Interface

Snapchat

What functionality is on the device?

What role does mobile phone play?



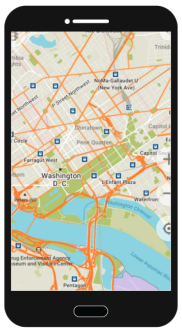
Apps

Most people's primary computing device

Snapchat

What functionality is NOT on the device?

What role does mobile phone play?



Apps

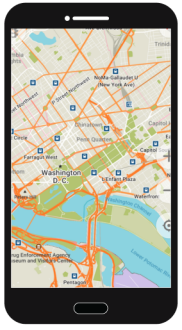
Most people's primary computing device

1. Peer-to-peer communication
2. User account storage
3. Friend location
4. Most "core" logic

Snapchat

What functionality is NOT on the device?

What role does mobile phone play?



Apps

Most people's primary computing device

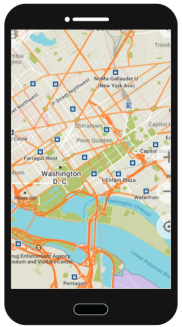
Snapchat

On Device: Window into application + sensors

Off Device: Main application

Main characteristic of an "App" is that it is an aggregation of multiple "services", not the main source of computation (or even code)

What role does mobile phone play?



Apps

Most people's primary computing device

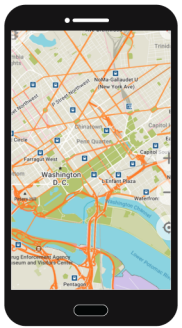
In this sense, "Apps" are primarily UI / UX programs

Snapchat

On Device: Window into application + sensors

Off Device: Main application

What role does mobile phone play?



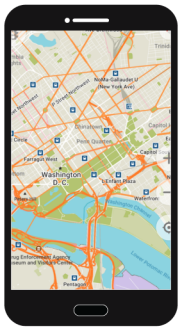
Games

Hearthstone

Game Engine / OpenGL

In this sense, mobile phone programmed more as a “traditional” computer

What role does mobile phone play?



Sensors

GPS

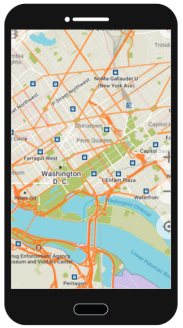
Accel.

Camera

Pokemon Go

Unique combination of “always on” + contextual information

What role does mobile phone play?



Sensors

GPS

Accel.

Camera

Unique combination of “always on” + contextual information

Augmented Reality +
Peer-to-peer Connectivity +
Gaming

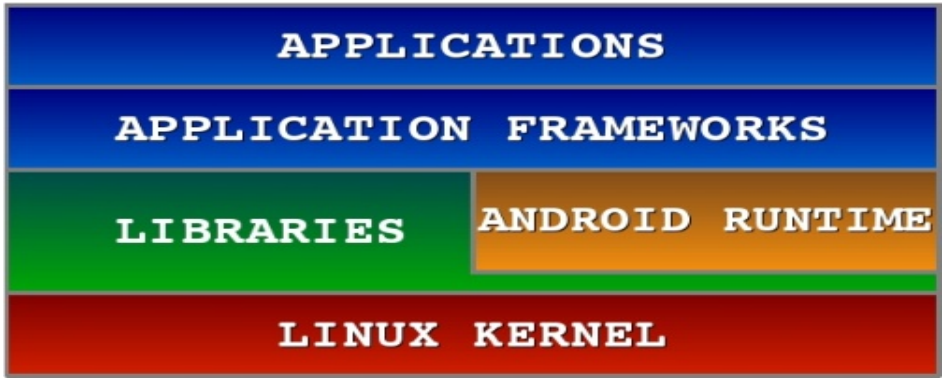
Pokemon Go

I feel this kind of use of Android is the “emerging platform”

What is Android?

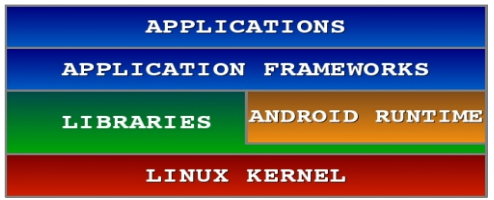
- Android is a software stack for mobile devices that includes an operating system, middleware and key applications.

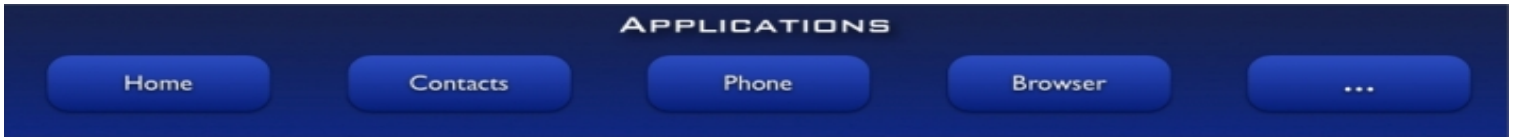
Android Architecture



- Android is a complete OS environment, and developing on it *conceptually* is no different than another other computer
- In reality, the nature of “application development” means that Android need not be so general. View it as Android “optimized” for app development.

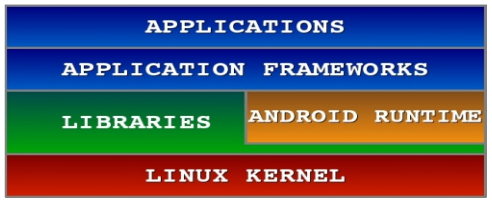
Applications





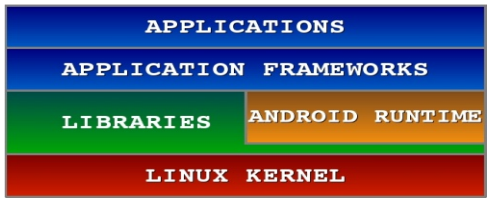
- Android provides a set of core applications, all written in Java:
  - ✓ Email Client
  - ✓ SMS Program
  - ✓ Calendar
  - ✓ Maps
  - ✓ Browser
  - ✓ Contacts
  - ✓ Etc

App Frameworks



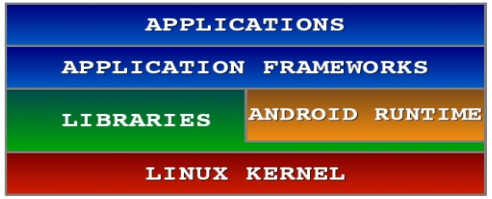
- Enabling and simplifying the reuse of components
  - ✓ Developers have full access to the same framework APIs used by the core applications.
  - ✓ Users are allowed to replace components.

Libraries



- Core functionality of libraries written in C/C++
  - GPS hardware interface
- Most of this functionality exposed to the application developer as a Java API (Application Frameworks) for app development
  - android.location
  - Google Location Services API

Run-time

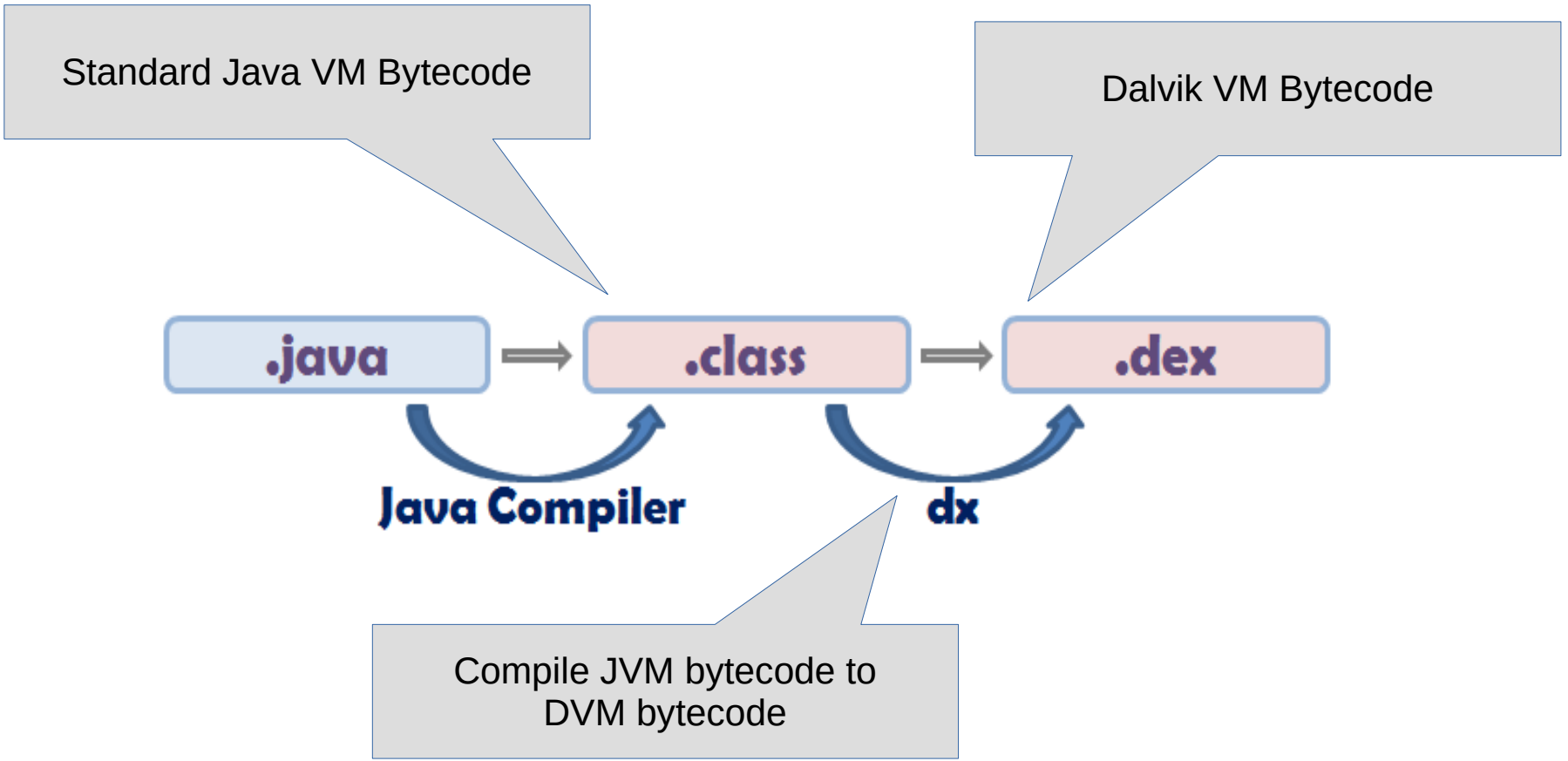


- Core Libraries
  - ✓ Providing most of the functionality available in the core libraries of the Java language
    - Data Structures, Utilities, File Access, Network Access, Graphics, etc
- Dalvik VM
  - ✓ Dalvik VM / Android Runtime (ART)

Dalvik VM

- Dalvik Virtual Machine
  - ✓ Each Android application runs in its own process, with its own instance of the Dalvik VM.
  - ✓ Dalvik has been written so that a device can run multiple VMs efficiently.
- The newer run-time system is called ART (we will revisit)
- Dalvik VM is not Java VM
  - ✓ JVM stack based, Dalvik register based
  - ✓ Dalvik optimized for mobile

Dalvik VM


Standard Java VM Bytecode
Dalvik VM Bytecode
.java
.class
.dex
Java Compiler
dx
Compile JVM bytecode to
DVM bytecode

Linux Kernel

- Relying on Linux Kernel 2.6 for core system services
- Providing an abstraction layer between the hardware and the rest of the software stack
- Not likely you'll have to work at this layer

General Development

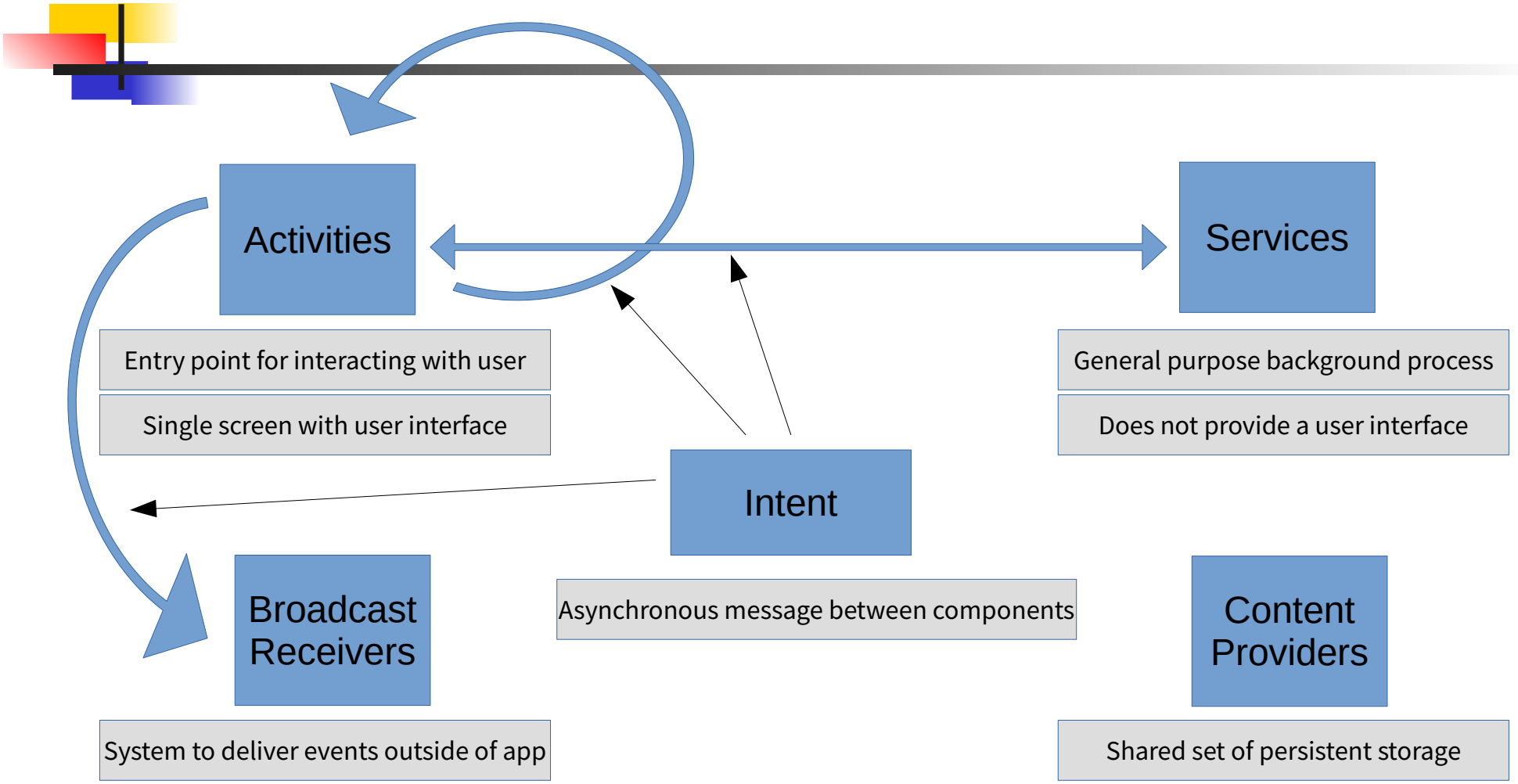
- Slightly “spoiled” from standard OS
  - When **development** and **deployment** are in the same environment, easy
  - When they are separate, there is always some kind of usability gap
    - Develop on linux machine, deploy to android device
  - Two approaches for android
    - Use an android simulator
    - Use an android device

General Development

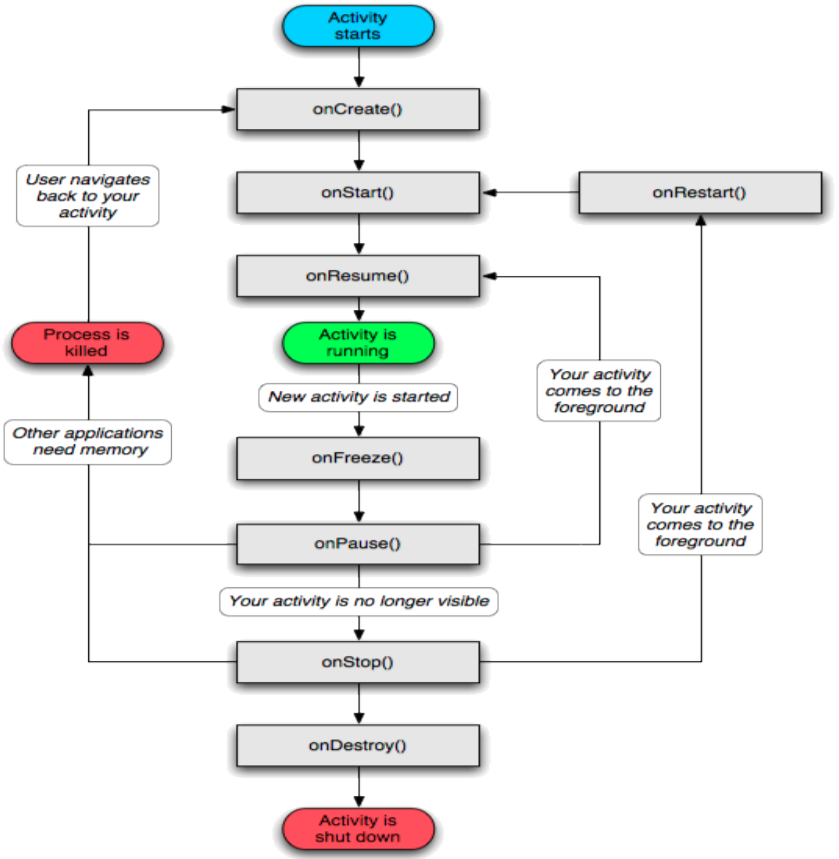
- Android Simulator
  - Pro: Low cost way to get started
  - Con: Not ideal for serious development (slow)
- Android Device
  - Pro: Working directly on hardware
  - Con: Likely need *at least* one development specific device

General Development

- Debugging
  - Part of the “usability gap”: Do you have println? Most of us debug using print statements
- Fortunately, android is a mature ecosystem with tools to aid this gap
  - android device bridge (adb)
  - adb logcat | grep “search”
    - Use android Log framework, or use print statements


Activities
Services
Entry point for interacting with user
Single screen with user interface
General purpose background process
Does not provide a user interface
Intent
Asynchronous message between components
Broadcast Receivers
System to deliver events outside of app
Content Providers
Shared set of persistent storage

Activity


Activity starts
onCreate()
onStart()
onRestart()
onResume()
Activity is running
User navigates back to your activity
Process is killed
Other applications need memory
New activity is started
Your activity comes to the foreground
Your activity comes to the foreground
onFreeze()
onPause()
Your activity is no longer visible
onStop()
onDestroy()
Activity is shut down

HelloAndroid.java

```
package com.example.helloandroid;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
      tv.setText("Hello,World. Hello, Android");
      setContentView(tv);
    }
}
```

Inherit from the Activity Class

Set the view “by hand” – from the program

Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.example.helloandroid"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".HelloAndroid"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>


</manifest>
```

the bootstrapping activity when the application starts

Shows in the "Application Launcher" so users can start it.

Activities and Views

- Each activity has a default window to draw in (although it may prompt for dialogs or notifications)
- The content of the window is a view or a group of views (derived from View or ViewGroup)
- Example of views: buttons, text fields, scroll bars, menu items, check boxes, etc.
- View(Group) made visible via Activity.setContentView() method.

Managing GUI Resources

```
<LinearLayout android:id="@+id/layout"

xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@color/white"
   android:orientation="vertical">

   <ScrollView
      android:id="@+id/scrollview"
      android:layout_width="match_parent"
      android:layout_height="0dp"
      android:layout_gravity="top"
      android:layout_weight="0.75">

      <TextView
         android:id="@+id/description"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_gravity="center"
         android:paddingBottom="@dimen/activity_vertical_margin"
         android:paddingLeft="@dimen/activity_horizontal_margin"
         android:paddingRight="@dimen/activity_horizontal_margin"
         android:paddingTop="@dimen/activity_vertical_margin"
         android:textColor="@color/white" />
   </ScrollView>
```

Define activity laytout + view in a resource xml file

Managing GUI Resources

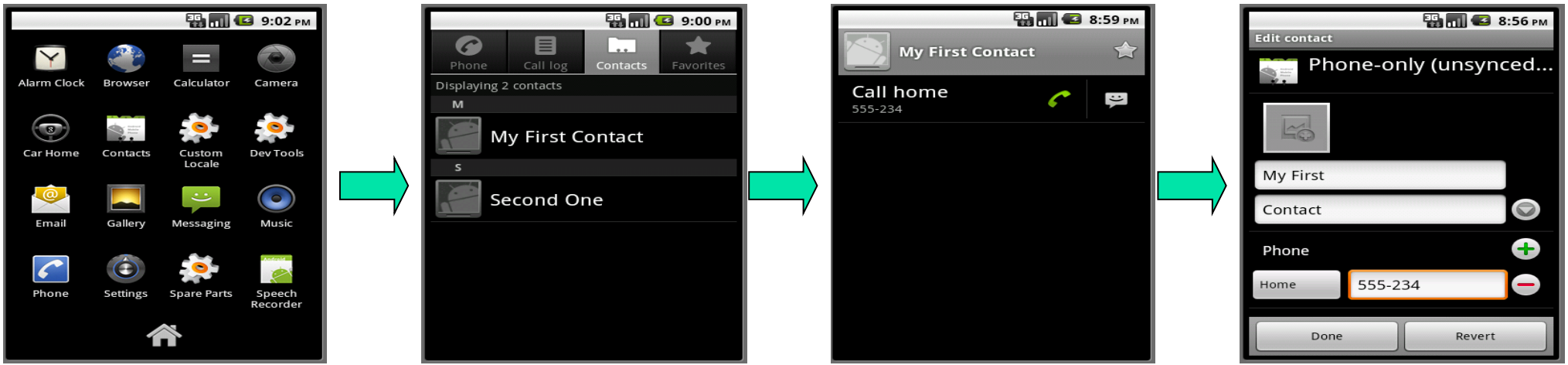
```
<LinearLayout android:id="@+id/layout"

xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@color/white"
   android:orientation="vertical">

   <ScrollView
      android:id="@+id/scrollview"
      android:layout_width="match_parent"
      android:layout_height="0dp"
      android:layout_gravity="top"
      android:layout_weight="0.75">
```

```
      <TextView
         android:id="@+id/description"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_gravity="center"
         android:paddingBottom="@dimen/activity_vertical_margin"
         android:paddingLeft="@dimen/activity_horizontal_margin"
         android:paddingRight="@dimen/activity_horizontal_margin"
         android:paddingTop="@dimen/activity_vertical_margin"
         android:textColor="@color/white" />
   </ScrollView>
```

Define activity laytout + view in a resource xml file

Why?

Activities start each other



most applications have multiple activities

Intents

- New activity may or may not live in the same application.
- If two activities live in different applications, you often do not know the name of the instantiated activity. Intent objects allow to describe “what you want” without naming it.
- Android’s elaborate intent resolution algorithm will find the appropriate activity for you. With principles and heuristics, “multi-matches” never exists eventually.

Explicit Intent

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

Implicit Intent

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT,
textMessage);
sendIntent.setType("text/plain");


startActivity(sendIntent);
```

Intent Filters

```
<?xml version="1.0" encoding="utf-8"?>
  <manifest . . . >
     <application . . . >
        <activity android:name="com.example.project.FreneticActivity"
                  android:icon="@drawable/small_pic.png"
                  android:label="@string/freneticLabel"
                  . . .  >
           <intent-filter . . . >
              <action android:name="android.intent.action.MAIN" />
              <category android:name="android.intent.category.LAUNCHER" />
           </intent-filter>
           <intent-filter . . . >
              <action android:name="com.example.project.BOUNCE" />
              <data android:mimeType="image/jpeg" />
              <category android:name="android.intent.category.DEFAULT" />
           </intent-filter>
        </activity>
        . . .
     </application>
  </manifest>
```

Handles JPEG images in some way

in general, intent filters give the activity the ability to say when it responds to activity creation request

Bundles in Intents

- As the “message” passed on to the instantiated activity, Intent objects can also be associated with extra information by the instantiating activity, in the form of “Bundle” object.
- Bundles can be placed in Intents through “putExtras” method (a variant of the “putExtra” method).

Built-in Intent Action Examples

Action	Usage
Intent.ACTION_BATTERY_LOW	The battery level has fallen below a threshold
Intent.ACTION_BATTERY_OKAY	The battery level has risen again
Intent.ACTION_HEADSET_PLUG	A headset was plugged in or a previously plugged headset was removed
Intent.ACTION_POWER_CONNECTED	The device has been plugged in
AudioManager.ACTION_AUDIO_BECOMING_NOISY	The internal audio speaker is about to be used instead of other output means (like a headset)

Building Block II: Service

- A Service is similar to an Activity, except that they run in the background like a Linux daemon with no GUI interfaces
- A service is started in similar ways such as an Activity: you need to manifest it, and it responds to Intents
- A service by default is started in the same thread of the rest of the App. (If you need multi-threading, you need to be explicitly creating threads, in ways similar to Java multi-threading)
- At execution time, services have lower priorities than (foreground) activities.

Building Block III: ContentProvider

- A ContentProvider (CP) manages data access and transfer across multiple Apps
- There are built-in CPs such as for Calendar and Contacts, but your App can provide your own CP
- CP can be used as a “lightweight database”, especially in scenarios where Calendar and Contact are queried

Data Export

- The exported data is in the form of tables
- Example: The built-in user dictionary provider might contain the following data:

Table 1: Sample user dictionary table.

word	app id	frequency	locale	_ID
mapreduce	user1	100	en_US	1
precompiler	user14	200	fr_FR	2
applet	user2	225	fr_CA	3
const	user1	255	pt_BR	4
int	user5	100	en_UK	5

Accessing CPs

- An application must create a ContentResolver (CR) client object to access data from a CP
- The relationship between a CP and a CR is the classic publish-subscribe pattern

- An App should request access to another App's CP via the manifest
  - E.g., access the built-in Contacts AP:

```
<uses-permission
android:name="android.permission.READ_CONTACTS"/>
```

- A CR object has methods that call same-named methods of a CP object

Accessing CPs

- Override the query method of the CR to access data
- public final Cursor **query** (Uri uri, String[] projection, String selection,String[] selectionArgs, String sortOrder)
  - uri = the URI that maps to the table in the provider
  - projection = the columns to be included for each retrieved row
  - selection = criteria for selecting rows

Example:

```
ContentResolver cr = getContentResolver();
Cursor cur =
  cr.query(ContactsContract.Contacts.CONTENT_URI, ..);
```

Example

```
ContentResolver cr = getContentResolver();
Cursor cur =
  cr.query(ContactsContract.Contacts.CONTENT_
  URI, null, null);

while (cur.moveToNext()) {
   String id =
     cur.getString(cur.getColumnIndex(
     ContactsContract.Contacts._ID));
}
```

Building Block IV: BroadCastReceiver

- A `BroadCastReceiver` can be veiwed as a lightweight Activity with no GUI interfaces.
- It is similar to an event handler in Java programming, whereas the events in this case are system intents

Building Block IV: BroadCastReceiver

- Manifest:

```
<receiver
    android:name=".ConnectivityChangeReceiver">
    <intent-filter>
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
    </intent-filter>
</receiver>
```

- Code:

```
class ConnnectivityChangeReceiver extends BroadcastReceiver{
    public void OnReceive(Context c, Intent i) { … }

}
```

Application Building Blocks Summary

- Activities – visual user interface focused on a single thing a user can do
- Services – no visual interface – they run in the background
- Content Providers – allow data exchange between applications
- Broadcast Receivers – receive and react to broadcast announcements

Starting/Shutting down components

- Activities
  - Start with “startActivity”
  - Can terminate itself via finish();
  - Can terminate other activities it started via finishActivity();
- Services
  - Start with “startService”
  - Can terminate via stopSelf(); or Context.stopService();
- Content Providers
  - Start when manifest matches
  - Are only active when responding to ContentResolvers
- Broadcast Receivers
  - Are only active when responding to broadcasts (start with “registerReceiver” and end with “unregisterReceiver”

Acknowledgments

- Mihail Sichitiu, “A Short Introduction to Android Programming”
- Gaydorus, Lubarsky, Powers, “Android programming with Broadcast Recievers, Services and Data Access”
- http://developer.android.com/resources/tutorials/hello-world.html
- https://github.com/googlesamples/android-MultiWindowPlayground