Introduction to Android and Android Application Development

1

Introduction to Android

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

2

What is OHA?

- **Quoting from *www.OpenHandsetAlliance.com* page**

- **"… Open Handset Alliance™, a group of 47 technology and mobile companies have come together to accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience.**
- **Together we have developed Android™, the first complete, open, and free mobile platform.**
- **We are committed to commercially deploy handsets and services using the Android Platform."**

3

What is Android?

- **Android™ consists of a complete set of software components for mobile devices including:**
  - an operating system,
  - middleware, and
  - embedded key mobile applications
  - a large market.
- **You can also say that the Android is a complete OS, not just a framework**

4

The Android Platform
again, what did they say about Android?

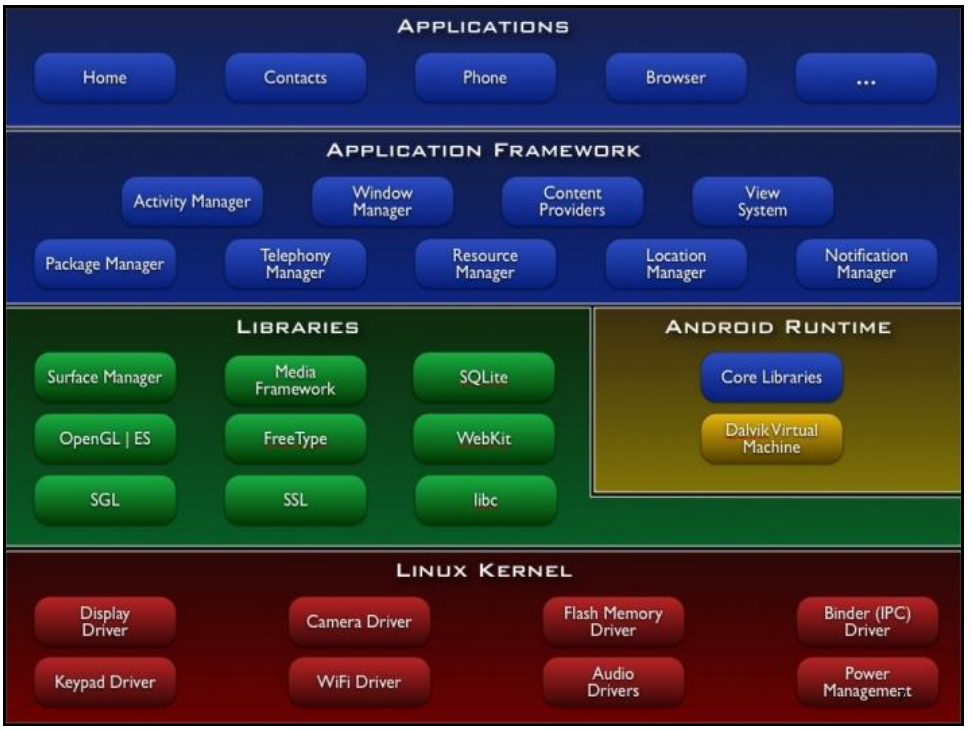
- **Android is a *software environment built for mobile devices.***
- **It is *not a hardware platform.***
- **Android includes:**
  - Linux kernel-based OS,
  - a rich UI,
  - telephone functionality,
  - end-user applications,
  - code libraries,
  - application frameworks,
  - multimedia support, ...
- **User applications are built for Android in Java.**

5

Android Components

- **Application framework enabling reuse and replacement of components**
- **Dalvik virtual machine optimized for mobile devices**
- **Integrated browser based on the open source WebKit engine**
- **Optimized graphics powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES specification (hardware acceleration optional)**
- **SQLite for structured data storage**
- **Media support for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)**
- **GSM Telephony (hardware dependent)**
- **Bluetooth, EDGE, 3G, 4G, and Wi-Fi (hardware dependent)**
- **Camera, GPS, compass, and accelerometer (hardware dependent)**
- **Rich development environment including a device emulator, tools for debugging, memory and performance profiling, and a plug-in for the Eclipse IDE**

6



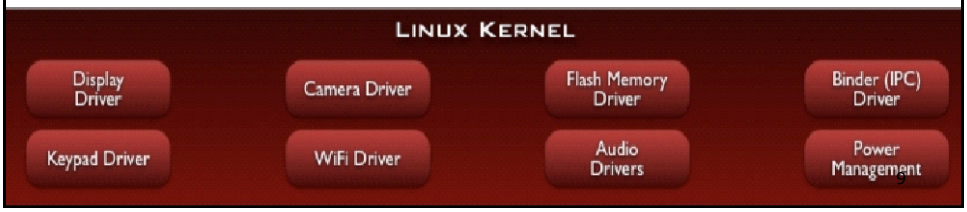
Why Linux for Phone?

- **Linux kernel is a proven core platform.**
- **Reliability is more important than performance when it comes to a mobile phone, because voice communication is the primary use of a phone.**
- **Linux provides a hardware abstraction layer, letting the upper levels remain unchanged despite changes in the underlying hardware.**
- **As new accessories appear on the market, drivers can be written at the Linux level to provide support, just as on other Linux platforms.**

8

Linux Kernel

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



9

Dalvik Virtual Machine

- **User applications, as well as core Android applications, are written in Java programming language and are compiled into *byte codes.***
- **Android *byte codes* are interpreted at runtime by a processor known as the *Dalvik virtual machine.***

10

Why another JVM?

- **Android bytecode files are logically equivalent to Java bytecodes, but they permit Android to**
  - run its applications in its own virtual environment that is free from Sun's licensing restrictions and
  - an open platform upon which Google, and potentially the open source community, can improve as necessary.

11

Dalvik Virtual Machine

- **Video (61 min)**
  - Available at: http://www.youtube.com/watch?v=ptjedOZEXPM

12

Android Architecture

- **Video 1/3: Android Architecture (13 min)**
  - Available at: http://www.youtube.com/watch?v=QBGfUs9mQYY
- **Video 2/3: Applications Life Cycle (8 min)**
  - Available at: http://www.youtube.com/watch?v=fL6gSd4ugSI&feature=channel
- **Video 3/3: Android's API (7 min)**
  - Available at: http://www.youtube.com/watch?v=MPukbH6D-lY&feature=channel

13

Android Application Framework

- **Video (52 min):**
  - Available at: http://sites.google.com/site/io/inside-the-android-application-framework

14

Development Tools

- Android SDK Tool

  http://developer.android.com/sdk/installing.html

  Download
  - If you run installer_r12-windows.exe, it will assess your system's environment.
  - If you run android-sdk_r12-windows.zip, unzip
    - Run Android SDK Tools Setup (remember the Android-sdk folder for eclipse)

  If your platform is Ubuntu

  see the URL:

  http://blog.sudobits.com/2011/07/27/android-sdk-for-ubuntu-11-0410-1011-10/

  If your platform is Fedora

  see the URL:

  http://www.if-not-true-then-false.com/2010/android-sdk-and-eclipse-adt-on-fedora-centos-red-hat-rhel/
- JDK – already installed, go to Android SDK Tool
  - –download it from java.oracle.com (http://www.oracle.com/technetwork/java/javase/downloads/jdk-6u25-download-346242.html)
  - –Install Java SE Development Kit 7.0
  - –set the PATH in Windows (System Properties > Advanced > Environment Variables > User variable PATH)

15

Development Tools

- AVD Manager (at the end of installation or in eclipse running SDK)
  - Virtual devices to be installed
  - Available packages to be installed (takes a while!!! – web access & install)
- Eclipse
  - http://www.eclipse.org/downloads/
    - Download & unzip
    - http://blog.sudobits.com/2011/04/24/how-to-install-eclipse-ide-on-ubuntu-11-04/
    - http://www.if-not-true-then-false.com/2010/linux-install-eclipse-on-fedora-centos-red-hat-rhel/
  - Run by setting the Workspace
    - can specify at every launch time
  - Run eclipse, Help > Install New Software > Add w/ FQAP (finding Android PATH)*
  - In eclipse, Windows > Preferences > Android > SDK location (find Android PATH)
  - File > New > Project > Android > Android Project > Next
  - Project Name: any; App Name: any; Package name: edu.lehman.cs.android.package1; Build Target
- Note:
  - Only if the above is unable; do the next slide

16

ADT

- Read the following carefully
  - http://developer.android.com/sdk/eclipse-adt.html#installing
- In eclipse,
  - Start Eclipse, then select Help > Install New Software....
  - Click Add, in the top-right corner.
  - In the Add Repository dialog that appears, in the location box enter following URL

    https://dl-ssl.google.com/android/eclipse/
  - Click OK
- Restart eclipse, make sure that the following is available:
  - Window > Android SDK & AVD Manager
  - File > New > Android Project

17

ADT

- **See the following document on the course web site**
  - AndroidSetupIntro.doc (based on vogella.de Gingerbread edition)
  - Or
  - http://www.vogella.de/articles/Android/article.html

18

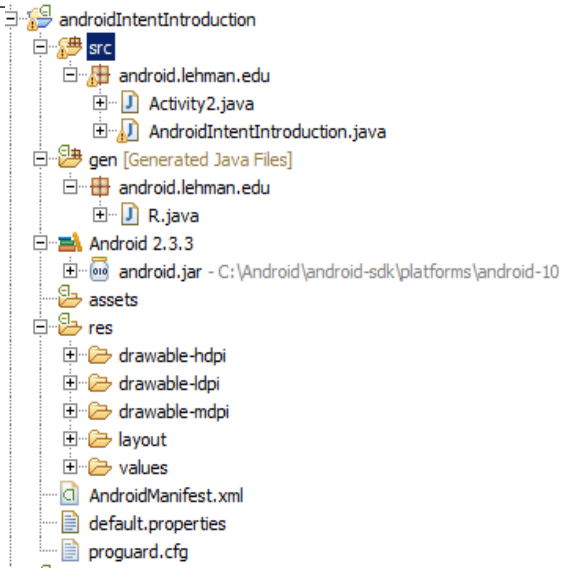
Android Applications

- **Each Android application runs in its own Linux process.**
- **An application consists of a combination of software components including:**
  - Activities
  - Services
  - Broadcast Receivers
  - Content Providers

19

Android Applications

- **Structure of a typical Android Application**



20

Android Application Life Cycle

- An Android application consists of one or more components that are defined in the application's manifest file. A component can be one of the following:
  - 1. An Activity
  - 2. A Service
  - 3. A broadcast receiver
  - 4. A content provider

21

Activity

- An activity usually presents a single visual user interface from which a number of actions could be performed.
- Although activities work together to form a cohesive user interface, each activity is independent of the others.
- Typically, one of the activities is marked as the first one that should be presented to the user when the application is launched.
- Moving from one activity to another is accomplished by having the current activity start the next one through so called **intents.**

22

Service

- A service doesn't have a visual user interface, but rather runs in the background for an indefinite period of time.
- It's possible to connect to (bind to) an ongoing service (and start the service if it's not already running).
- While connected, you can communicate with the service through an interface that the service exposes.

23

Broadcast receiver

- A broadcast receiver is a component that does nothing but receive and react to broadcast announcements.
- Many broadcasts originate in system code (e.g., “you got mail“) but any other applications can also initiate broadcasts.
- Broadcast receivers do not display a user interface. However, they may start an activity in response to the information they receive, or -as services do -they may use the notification manager to alert the user.

24

Content provider

- A content provider makes a specific set of the application's data available to other applications.
- The data usually is stored in the file system, or in an SQLite database.
- The content provider implements a standard set of methods that enable other applications to retrieve and store data of the type it controls.
- However, applications do not call these methods directly. Rather they use a content resolver object and call its methods instead. A content resolver can talk to any content provider; it cooperates with the provider to manage any interprocess communication that's involved.

25

Android Applications

- **Every Android application runs in its own process** (with its own instance of the Dalvik virtual machine).
- Whenever there's a request that should be handled by a particular component,

  Android makes sure
  - that the application process of the component is running, starting it if necessary, and
  - that an appropriate instance of the component is available, creating the instance if necessary.

26

Application's Life Cycle

- A Linux process encapsulating an Android application is created for the application when some of its code needs to be run, and will remain running until
  - 1. it is no longer needed, OR
  - 2. the system needs to reclaim its memory for use by other applications.

27

Application's Life Cycle

- **An unusual and fundamental feature of Android is that an application process's lifetime is not directly controlled by the application itself.**
- Instead, it is determined by the system through a combination of
  - 1. the parts of the application that the system knows are running,
  - 2. how important these things are to the user, and
  - 3. how much overall memory is available in the system

28

Component Lifecycle

- Application components have a **lifecycle**
  - 1. A beginning when Android instantiates them to respond to intents
  - 2. An end when the instances are destroyed.
  - 3. In between, they may sometimes be active or inactive, or -in the case of activities- visible to the user or invisible.

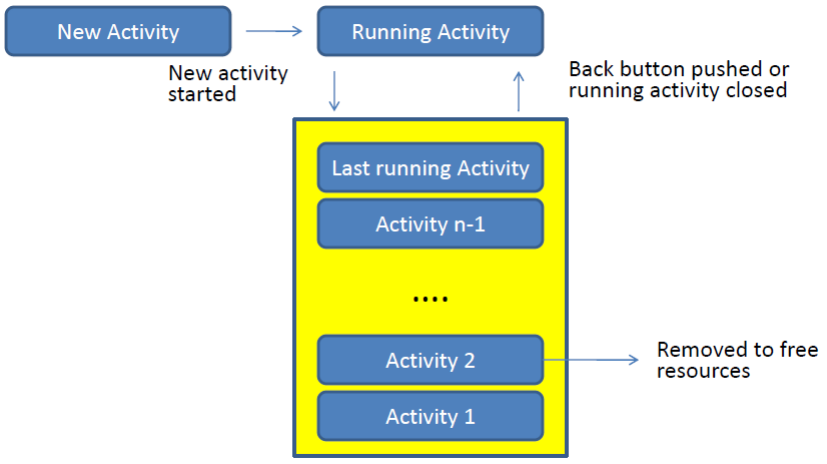
29

Activity Stack

- Activities in the system are managed as an **activity stack.**
- When a new activity is started, it is placed on the top of the stack and becomes the running activity --the previous activity always remains below it in the stack, and will not come to the foreground again until the new activity exits.
- If the user presses the Back Button the next activity on the stack moves up and becomes active.

30

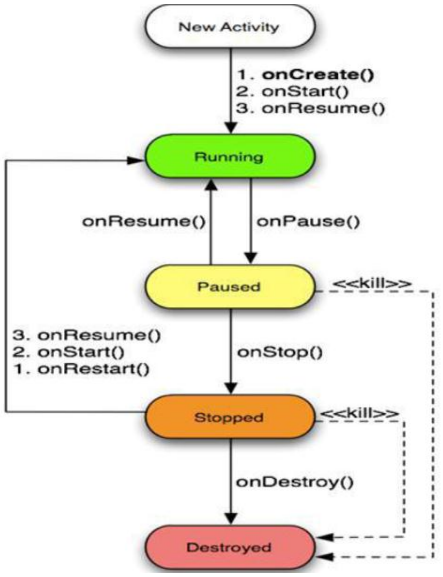
Activity Stack



31

Lifecycle States

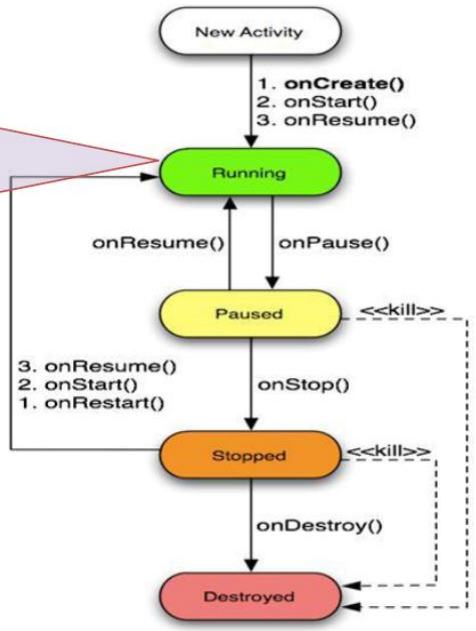
- An activity has essentially three states:
  1. It is active or running
  2. It is paused or
  3. It is stopped.



32

It is ***active or running*** when it is in the *foreground* of the screen(at the top of the activity stack for the current task).

This is the activity that is the focus for the user's actions.

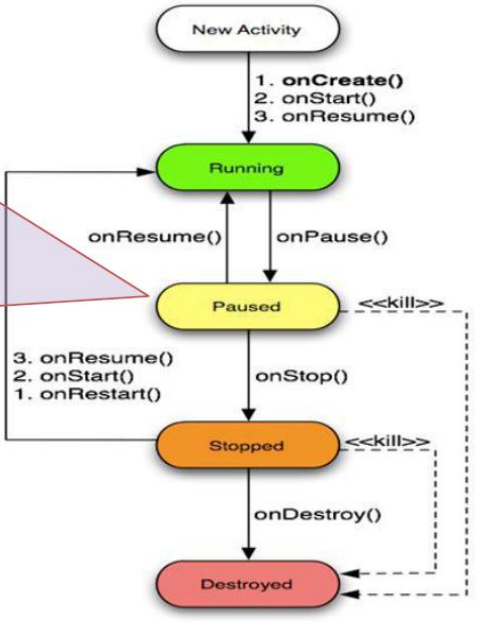


33

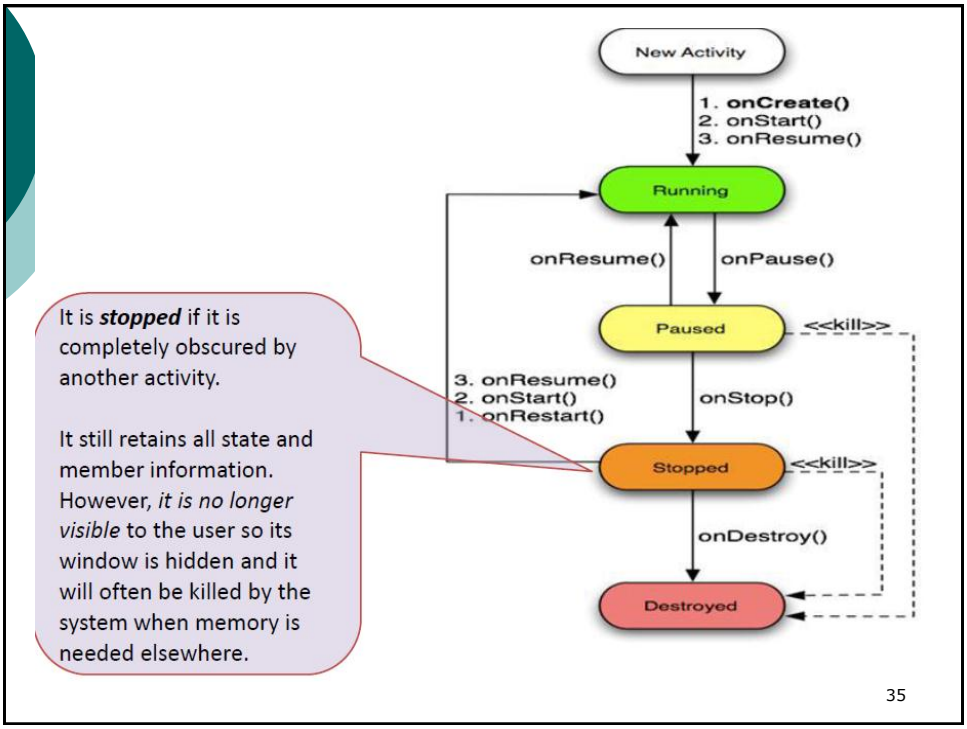
It is **paused** if it has lost focus but is still visible to the user.

That is, another activity lies on top of it and that new activity either is transparent or doesn't cover the *full screen*.

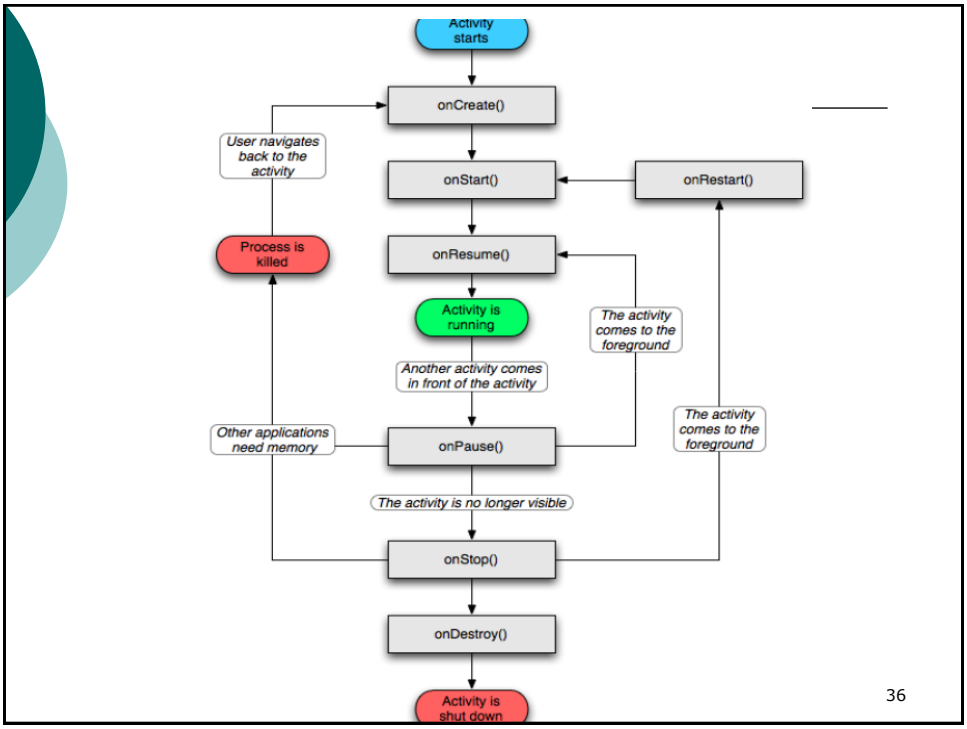
A paused activity is completely *alive* (it maintains all state and member information and remains attached to the window manager), but can be killed by the system in extreme low memory situations.



34


New Activity
1. onCreate()
2. onStart()
3. onResume()
Running
onResume()
onPause()
Paused
<<kill>>
3. onResume()
2. onStart()
1. onRestart()
onStop()
Stopped
<<kill>>
onDestroy()
Destroyed
It is ***stopped*** if it is completely obscured by another activity.
It still retains all state and member information. However, *it is no longer visible* to the user so its window is hidden and it will often be killed by the system when memory is needed elsewhere.

35


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

36

Android XML & User Interface

- **Basic XML layouts**
  - Will be studied based on a few examples

Layouts (e.g., Linearlayout)

http://developer.android.com/guide/topics/ui/layout-objects.html

37

Android Widgets

Studied based on examples

38

Inside Android: Intents

- **An important and recurring theme of Android development is the Intent.**
- **An Intent in Android describes *what you want to do.***
- **This may look like**
  - *"I want to look up a contact record," or*
  - *"Please launch this website," or*
  - *"Show the Order Confirmation Screen."*
- **Intents are important because they facilitate navigation and represent the most important aspect of Android coding.**

39

Intents & IntentFilters

- **An Intent is a declaration of need.**
- **An Intent is made up of various pieces including:**
  - desired *action or service,*
  - *data, and*
  - *category of component that should handle the intent and instructions on how to launch a target activity.*
- **An IntentFilter is a trigger, a declaration of capability and interest in offering assistance to those in need.**
- **An IntentFilter may be generic or specific with respect to which Intents it offers to service.**

40

Intents & IntentFilters

- **An intent is an abstract description of an operation to be performed.**
- **Its most significant use is in the *launching of activities*, where it can be thought of as the glue between activities.**
- **The primary pieces of information in an intent are:**

Action	Data
The general action to be performed, such as: ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, etc.	The data to operate on, such as a person record in the contacts database, expressed as a Uri.

41

Intents & IntentFilters

Some examples of Intent's action/data pairs are:

- **ACTION_VIEW *content://contacts/1--Display information about the person whose identifier is "1".***
- **ACTION_DIAL *content://contacts/1--Display the phone dialer with the person filled in.***
- **ACTION_VIEW*tel:123--Display the phone dialer with the given number filled in***
- **ACTION_DIAL *tel:123--Display the phone dialer with the given number filled in.***
- **ACTION_EDIT *content://contacts/1--Edit information about the person whose identifier is "1".***
- **ACTION_VIEW *content://contacts/--Display a list of people, which the user can browse through.***

42

Dissecting Intents

1. **Component name The name of the component that should handle the intent (for example "com.example.project.app.MyActivity1").**
2. **Action A string naming the action to be performed —or, in the case of broadcast intents, the action that took place and is being reported (for example: ACTION_VIEW, ACTION_CALL, ACTION_TIMEZONE_CHANGED, …).**
3. **Data The URI of the data to be acted on and the MIME type of that data (for example tel:/718-960-1234 , "http://maps.google.com", ...).**
4. **Category A string containing additional information about the kind of component that should handle the intent (for example CATEGORY_BROWSABLE, CATEGORY_LAUNCHER, …).**
5. **Extras *Key-value pairsfor additional information that should be delivered to the component handling the intent.***
6. **Flags of various sorts.**

43

Intent: Example

- **Following Intent uses built-in task (*ACTION_VIEW*) to explore a web page**
- **(see new Uri value)**

```
Intent myIntent= new Intent(
   Intent.ACTION_VIEW,

Uri.parse("http://www.google.com"));
startActivity(myIntent);
```

44

Delivering Intents

- An Intent object is passed to Context.startActivity() or Activity.startActivityForResult() to launch an activity or get an existing activity to do something new (*asynchronous & synchronously* respectively).
- An Intent object is passed to Context.startService() to initiate a service or deliver new instructions to an ongoing service.
- An intent can be passed to Context.bindService() to establish a connection between the calling component and a target service. It can optionally initiate the service if it's not already running.

45

Intent Resolution

Intents can be divided into two groups:

- *Explicit intents* designate the target component by its name, typically used for an activity starting a subordinate service or launching a sister activity.
- *Implicit intents* do not name a target (the field for the component name is blank). Implicit intents are often used to activate components in other applications. Late binding applies.

Whenever possible Android delivers an explicit intent to an instance of the designated target class.

46

Intent: Example

- **Following fragments calls an Intent whose job is to invoke a built-in task (*ACTION_VIEW*) and explore the *Contacts* available in the phone.**

```
Intent myIntent= new Intent (
        Intent.ACTION_VIEW,
        Uri.parse("content://contacts/people"));

startActivity(myIntent);
```

Try in Android Emulator

47

Intent: Example

- **Following Intent uses built-in task (*ACTION_VIEW*) to explore a web page**
- **(see new Uri value)**

```
Intent myIntent= new Intent(
   Intent.ACTION_VIEW,

Uri.parse("http://www.google.com"));
startActivity(myIntent);
```

48

Intent: Example

- **Following Intent uses built-in task (*ACTION_VIEW*) to make a phone call**
- **(see new Uri value)**

```
Intent myIntent= new Intent(
   Intent.ACTION_VIEW,
      Uri.parse("tel:/914 674-7461"));
startActivity(myIntent);
```

49

IntentFilters (Revisted)

- **The IntentFilter defines the relationship between the Intent and the application.**
- **IntentFilters can be specific to the data portion of the Intent, the action portion, or both.**
- **IntentFilters also contain a field known as a *category.* A *category helps classify the action.***
- **For example, the category named**

  CATEGORY_LAUNCHER

  instructs Android that the Activity containing this IntentFilter should be visible in the home screen.

50

More about Android Intents

Intents

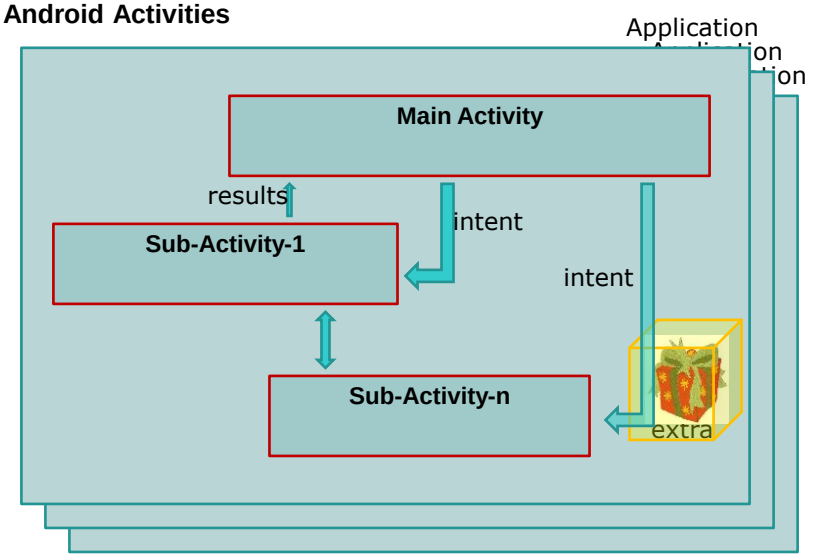
Android Activities

An Android application could include any number of activities.

- **An *activity* uses the *setContentView(...)* method to expose (usually) a single UI from which a number of actions could be performed.**
- **Activities are independent of each other; however they usually cooperate exchanging data and actions.**
- **Typically, one of the activities is designated as the first one (*main)* that should be presented to the user when the application is launched.**
- **Moving from one activity to another is accomplished by asking the current activity to execute an *intent*.**
- **Activities interact with each other in an asynchronous mode.**

Intents

- **Android Activities**



Intents

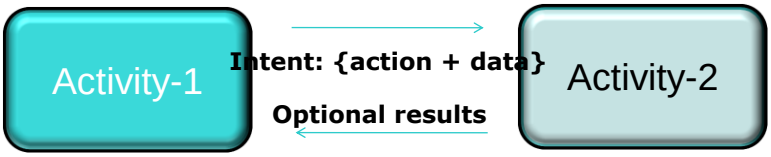
- **Intents are invoked using the following options**

startActivity(intent)	launches an *Activity*
sendBroadcast(intent)	sends an intent to any interested *BroadcastReceiver* components
startService(intent) ***or*** ***bindService(intent, …)***	communicate with a background Service.

Intents

The main arguments of an Intent are:

1. **Action** The built-in action to be performed, such as ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, …or *user-created-activity*
2. **Data** The primary data to operate on, such as a phone number to be called (expressed as a Uri).



Intents

- **Typically an intent is called as follows:**

```
Intent myActivity = new Intent (action,
                                data);

startActivity(myActivity);
```

Primary data (as an URI)
tel://
http://
sendto://

Built-in or user-created activity

Example: Intents

Examples of action/data pairs are:

- **ACTION_DIAL** ***tel:123***
  - Display the phone dialer with the given number filled in.
- ***ACTION_VIEW*** ***http://www.google.com***
  - Show Google page in a browser view. Note how the VIEW action does what is considered the most reasonable thing for a particular URI.
- ***ACTION_EDIT*** ***content://contacts/people/2***
  - Edit information about the person whose identifier is "2".
- ***ACTION_VIEW*** ***content://contacts/people/2***
  - Used to start an activity to display 2ndperson.
- ***ACTION_VIEW*** ***content://contacts/people/***
  - Display a list of people, which the user can browse through. Selecting a particular person to view would result in a new intent

Intents

Built-in Standard Actions

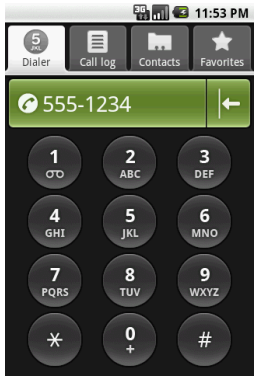
List of standard actions that Intents can use for launching activities (usually through *startActivity(Intent).*

ACTION_MAIN	ACTION_ANSWER
ACTION_VIEW	ACTION_INSERT
ACTION_ATTACH_DATA	ACTION_DELETE
ACTION_EDIT	ACTION_RUN
ACTION_PICK	ACTION_SYNC
ACTION_CHOOSER	ACTION_PICK_ACTIVITY
ACTION_GET_CONTENT	**ACTION_SEARCH**
ACTION_DIAL	**ACTION_WEB_SEARCH**
ACTION_CALL	ACTION_FACTORY_TEST
ACTION_SEND	
ACTION_SENDTO	

Example: Intents

- **Display the phone dialer with the given number filled in.**

```
Intent myActivity2 = new Intent(Intent.ACTION_DIAL,
                                Uri.parse( "tel:555-1234"));
startActivity(myActivity2);
```

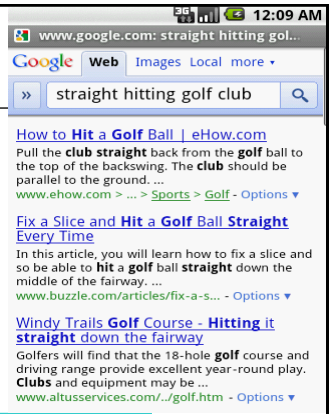


Intents

Intents -Secondary Attributes

- **In addition to the primary *action/data* attributes, there are a number of secondary attributes that you can also include with an intent, such as:**

  1.Category2. Components3. Type4. Extras



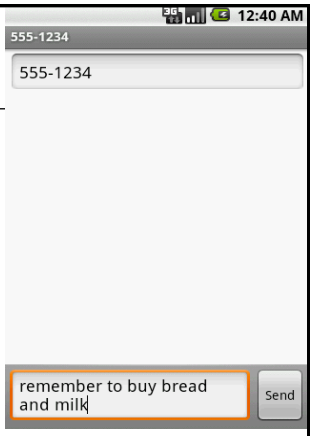
Example: Doing a Google search looking for golf clubs

```
Intent intent= new Intent (Intent.ACTION_WEB_SEARCH);
intent.putExtra(SearchManager.QUERY, "straight hitting golf
clubs");
startActivity(intent);
```

Secondary data

Intents

Intents -Secondary Attributes



Example: Sending a text message (using extra attributes)

```
Intent intent= new Intent( Intent.ACTION_SENDTO, Uri.parse ("sms:5551234"));
intent.putExtra("sms_body", “remember to buy bread and milk”);
startActivity(intent);
```

Intents

Intents -Secondary Attributes

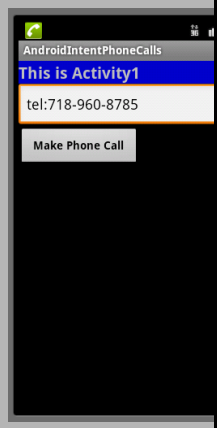
Example: **Showing Pictures (using extra attributes)**

```
Intent myIntent= new Intent();
myIntent.setType("image/pictures/*");
myIntent.setAction(Intent.ACTION_GET_CONTENT);
startActivity(myIntent);
```

Example Intents

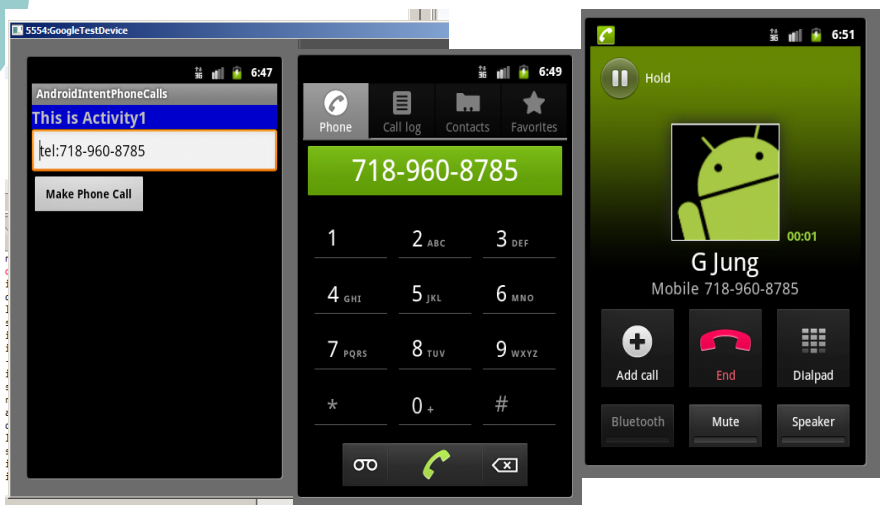
1. A Complete Example: Activity1 displays an interface to accept a phone number and requests (built-in) Activity2 to make the call.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent" >
    <TextView android:id="@+id/label1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff0000cc"
android:text="This is Activity1"
android:textStyle="bold"
android:textSize="20sp" />
    <EditText android:id="@+id/text1"
android:layout_width="fill_parent"
android:layout_height="54px"
android:text="tel:718-960-8785"
android:textSize="18sp" />
   <Button android:id="@+id/btnCallActivity2"
android:layout_width="149px"
android:layout_height="wrap_content"
android:text="Make Phone Call"
```



Intents

1. A Complete Example: Activity1 displays an interface to accept a phone number and requests (built-in) Activity2 to make the call.



Example Intents

1. A Complete Example: Activity1 displays an interface to accept a phone number and requests (built-in) Activity2 to make the call.

```
package edu.lehman.cs.intents;
import android.app.Activity;
import android.os.Bundle;
import android.content.Intent;
import android.net.Uri;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.*;

public class AndroidIntentPhoneCallsActivity extends Activity {
    TextView label1;
    EditText text1;
    Button btnCallActivity2;
    @Override
    public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    try{
 setContentView(R.layout.main);
 label1= (TextView)findViewById(R.id.label1);
 text1= (EditText)findViewById(R.id.text1);
 btnCallActivity2= (Button)findViewById(R.id.btnCallActivity2);
```

Example Intents

1. A Complete Example: Activity1 displays an interface to accept a phone number and requests (built-in) Activity2 to make the call.

```
btnCallActivity2.setOnClickListener(new ClickHandler());
        } catch(Exception e) {
   Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_LONG).show();
        }
    } //onCreate
 private class ClickHandler implements OnClickListener {
   @Override
   public void onClick(View v) {
   try {
   // myActivity2 places a phone call for ACTION_CALL or ACTION_DIAL
   // use 'tel:' formatted data: "tel:718-960-8785"
   // for ACTION_VIEW use data: "http://www.youtube.com"
   // (you also need INTERNET permission -see Manifest)
   String myData= text1.getText().toString();
   Intent myActivity2 = new Intent(Intent.ACTION_DIAL,  Uri.parse(myData));
   startActivity(myActivity2);
       } catch(Exception e) {
   Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_LONG).show();
       }
   }//onClick
   }//ClickHandler
  }
```

Example Intents

1. A Complete Example: Activity1 displays an interface to accept a phone number and requests (built-in) Activity2 to make the call.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.lehman.cs.intents"
    android:versionCode="1"
    android:versionName="1.0">
<application android:icon="@drawable/icon"
   android:label="@string/app_name">
<activity  android:name=".AndroidIntentPhoneCallsActivity"
 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>
<uses-sdk android:minSdkVersion="10" />
</manifest>
```

Intents

Built-in Standard Broadcast Actions

List of standard actions that Intents can use for receiving broadcasts (usually through *registerReceiver(BroadcastReceiver, IntentFilter)* or a <receiver>tag in a manifest).

ACTION_TIME_TICK
ACTION_TIME_CHANGED
ACTION_TIMEZONE_CHANGED
ACTION_BOOT_COMPLETED
ACTION_PACKAGE_ADDED
ACTION_PACKAGE_CHANGED
ACTION_PACKAGE_REMOVED
ACTION_UID_REMOVED
ACTION_BATTERY_CHANGED

Intents

More Examples: Using Standard Actions

Call Immediately

Modify the *complete* example1 replacing the method 'ClickHandler' with the following code

```
String myData= "tel:718-960-8785";
Intent myActivity2 = new Intent(Intent.ACTION_CALL,
Uri.parse(myData));
startActivity(myActivity2);
```

Needs Permission:
<uses-permission android:name="android.permission.CALL_PHONE" />

Intents

More Examples: Using Standard Actions

Show all your Contacts

Modify the *complete* example1 replacing the method 'ClickHandler' with the following code

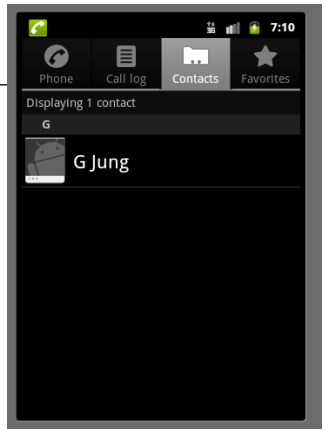
```
String myData= "content://contacts/people/";
Intent myActivity2 = new Intent(Intent.ACTION_VIEW,
Uri.parse(myData));
startActivity(myActivity2);
```

Intents

More Examples: Using Standard Actions

Show a Particular Contact (ID = 2)

Modify the *complete* example1 replacing the method 'ClickHandler' with the following code




```
String myData= "content://contacts/people/2";
Intent myActivity2 = new Intent(Intent.ACTION_VIEW,
Uri.parse(myData));
startActivity(myActivity2);
```

Intents

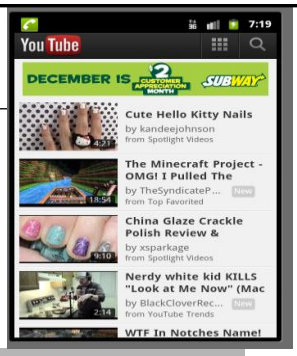
More Examples: Using Standard Actions

Edit a Particular Contact (ID = 2)

Modify the *complete* example1 replacing the method 'ClickHandler' with the following code

```
String myData= "content://contacts/people/2";
Intent myActivity2 = new Intent(Intent.ACTION_EDIT,
Uri.parse(myData));
startActivity(myActivity2);
```

Intents



More Examples: Using Standard Actions

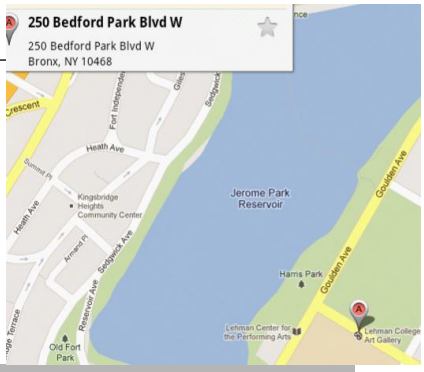
View a Webpage

Modify the *complete* example1 replacing the method 'ClickHandler' with the following code

```
String myData= "http://www.youTube.com";
Intent myActivity2 = new Intent(Intent.ACTION_VIEW,
Uri.parse(myData));
startActivity(myActivity2);
```

Caution. Add to the Manifest a request to use the Internet:
<uses-permission
android:name="android.permission.INTERNET" />

Intents



More Examples: Using Standard Actions

Geo Mapping an Address

Provide a geoCode expression holding a street address (Replace spaces with '+')

```
String geoCode=
"geo:0,0?q=250+west+Bedford+Park+Blvd+West+Bronx+NY";
Intent myIntent = new Intent(Intent.ACTION_VIEW,
Uri.parse(geoCode));
startActivity(myIntent);
```

Modify the Manifest adding the following requests:
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.INTERNET"/>

Intents

More Examples: Using Standard Actions

Geo Mapping Coordinates (latitude, longitude)

Provide a geoCodeholding latitude and longitude (also an addittional zoom '?z=xx' with xx in range 1..23)


```
String geoCode= "geo:41.5020952,-81.6789717";
Intent myActivity2 = new Intent(Intent.ACTION_VIEW, Uri.parse(myData));
startActivity(myActivity2);
```

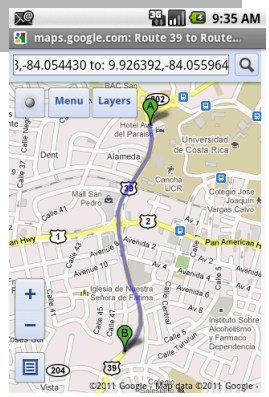
Modify the Manifest adding the following requests:

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

Intents

More Examples:

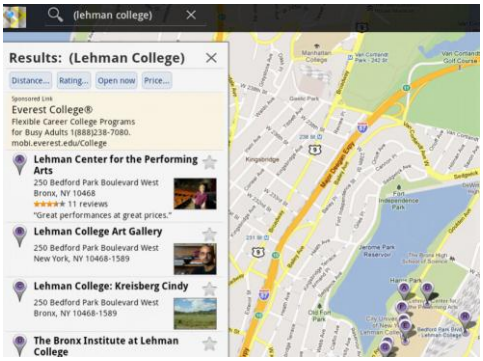
```
// Getting driving directions: how to go from loaction A to location B?
Intent intent = new Intent(android.content.Intent.ACTION_VIEW,
Uri.parse("http://maps.google.com/maps?saddr=9.938083,-84.054430&daddr=9.926392,-84.055964"));
startActivity(intent);
```



Intents

More Examples:

```
// use a mnemonic to articulate an address
String thePlace= "Lehman College";
Intent intent= new Intent(android.content.Intent.ACTION_VIEW,
Uri.parse("geo:0,0?q= (" + thePlace+ ")" );
startActivity(intent);
```

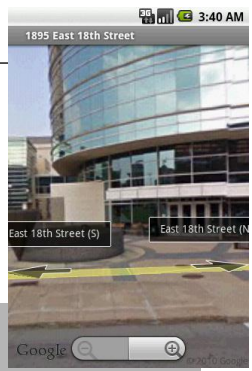


Intents

More Examples: Using Standard Actions

Geo Mapping – Google StreetView

geoCodeUri structure:
google.streetview:cbll=*lat,lng*&cbp=1,
yaw,,pitch,zoom&mz=mapZoom



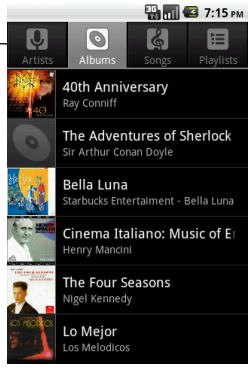

```
String geoCode=
"google.streetview:cbll=41.5020952,-
81.6789717&cbp=1,270,,45,1&mz=1";
Intent intent= new Intent(Intent.ACTION_VIEW,
Uri.parse(geoCode));
startActivity(intent);
```

Modify the Manifest adding the following requests:
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.INTERNET"/>

Intents

More Examples: Using Standard Actions

Launching the Music Player

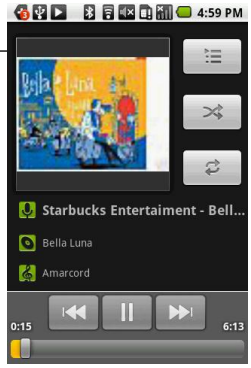



```
//launch music player
Intent myActivity2 =  new Intent("android.intent.action.MUSIC_PLAYER");
startActivity(myActivity2);
```

Intents

More Examples: Using Standard Actions

Playing a song stored in the SD card



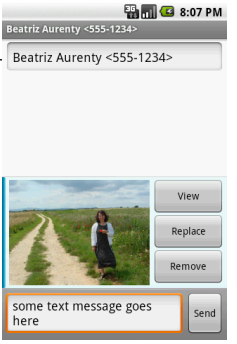

```
// play song "amarcord.mp3" saved in the SD
Intent myActivity2 = new
Intent(android.content.Intent.ACTION_VIEW);
Uri data = Uri.parse("file:///sdcard/amarcord.mp3");
String type = "audio/mp3";
myActivity2.setDataAndType(data, type);
startActivity(myActivity2);
```

Intents

More Examples: Using Standard Actions

Sending MMS

Add picture #1 from SD to MMS



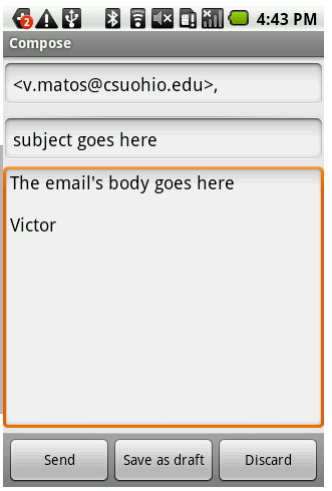

```
//send mms attach picture #1 to it
Uri uri= Uri.parse("content://media/external/images/media/1");
myActivity2 = new Intent(Intent.ACTION_SEND);
myActivity2.putExtra("address", "555-1234");
myActivity2.putExtra("sms_body", "some text message goes here");
myActivity2.putExtra(Intent.EXTRA_STREAM, uri);
myActivity2.setType("image/png");
startActivity(myActivity2);
```

Intents

More Examples: Using Standard Actions

Sending Email

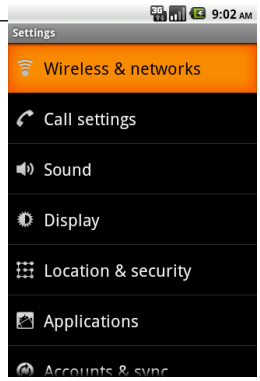
```
// send email
Uri uri= Uri.parse("mailto:v.a@csuohio.edu");
Intent myActivity2 = new
Intent(Intent.ACTION_SENDTO, uri);
// you may skip the next two pieces [subject/text]
myActivity2.putExtra(Intent.EXTRA_SUBJECT,
        "subject goes here");
myActivity2.putExtra(Intent.EXTRA_TEXT,
        "The email's body goes here");
startActivity(myActivity2);
```



Intents

More Examples: Using Standard Actions

Setting System



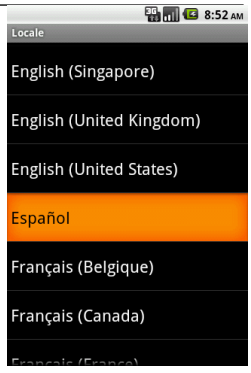

```
Intent intent= new Intent(
android.provider.Settings.ACTION_SETTINGS);
startActivity(intent);
```

Intents

More Examples: Using Standard Actions

Setting System Locale:

Language & Keyboard




```
Intent intent= new Intent(
android.provider.Settings.ACTION_LOCALE_SETTINGS);
startActivity(intent);
```

Intents

Starting Activities and Getting Results

- **The startActivity(Intent) method is used to start a new activity, which will be placed at the top of the activity stack.**

Sometimes you want to get a result back from the called sub-activity when it ends.

- **For example, you may start an activity that let the user pick a person from a list of contacts; when it ends, it returns the person that was selected.**

Intents

Starting Activities and Getting Results

In order to get results back from the called activity we use the method

```
startActivityForResult( Intent, requestCodeID)
```

Where the second (*requestCodeID)* parameter identifies the call.

The result sent by the sub-activity could be picked up through the asynchronous method

```
onActivityResult( requestCodeID, resultCode, Intent )
```

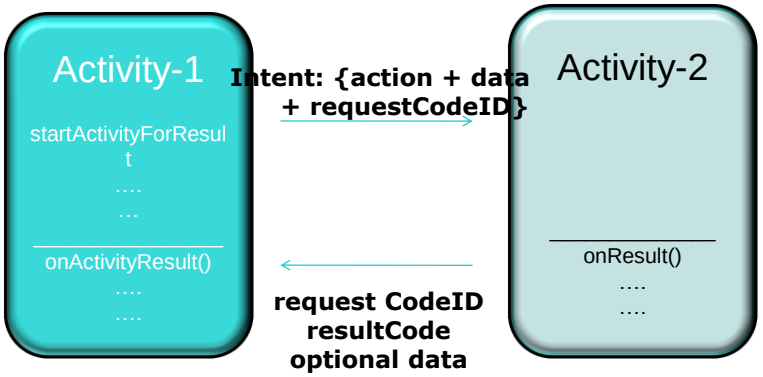
Intents

Starting Activities and Getting Results

- Before an activity exits, it can call **setResult(resultCode)** to return a termination signal back to its parent.
- Always supply a result code, which can be the standard results **Activity.RESULT_CANCELED**, **Activity.RESULT_OK**,
  or any custom values.
- All of this information can be capture back on the parent's onActivityResult (int requestCodeID, int resultCode, Intent data) along with the integer identifier it originally supplied.
- If a child activity fails for any reason (such as crashing), the parent activity will receive a result with the code **RESULT_CANCELED**.

Intents

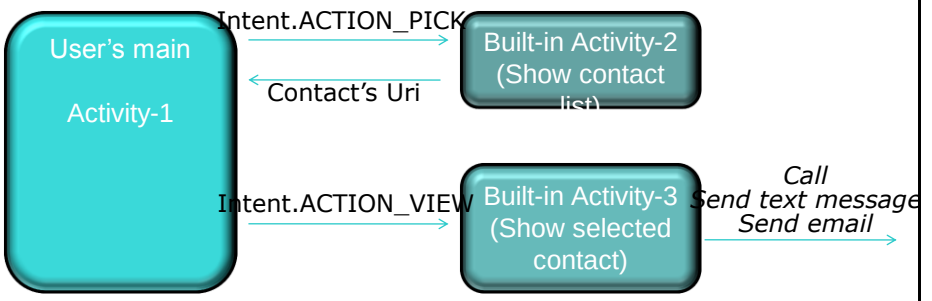
Starting Activities and Getting Results



Intents

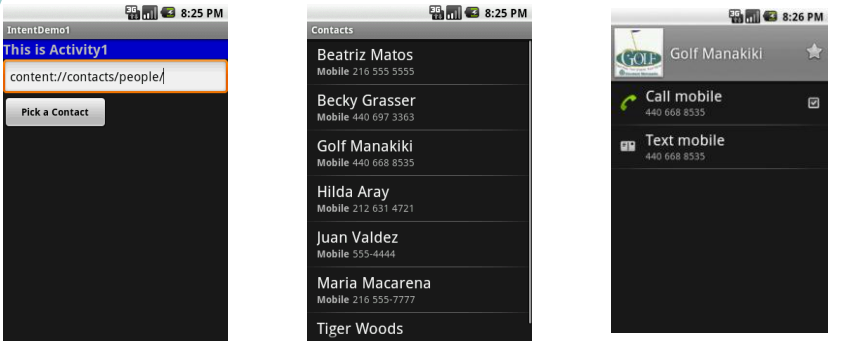
Example. Let's play golf -Call for a tee-time.

1. Show all contacts and pick a particular one (*Intent.ACTION_PICK).*
2. For a successful interaction the main-activity accepts the returned URI identifying the person we want to call (*content://contacts/people/n).*
3. 'Nicely' show the selected contact's entry allowing calling, texting, emailing actions (Intent.ACTION_VIEW).



Example: Golf Call

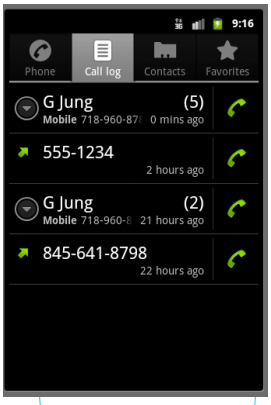
Example2. Let's play golf -*Call for a tee-time.*

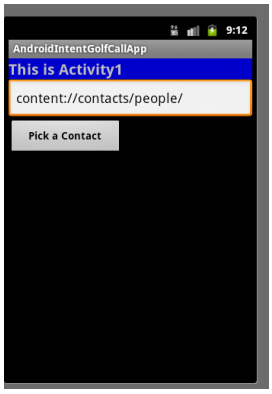


Main activity Intent.ACTION_PICK Intent.ACTION_VIEW

Example 2: Intents

Example2. Let's play golf -*Call for a tee-time.*





Place the call

Terminate the call

Selected Content's Uri

Example 2: Intents

Example2. *Calling a sub-activity, receiving results.*

```
//Intent: making a phone call
//receiving results from a sub-activity
package edu.lehman.cs.android.intents;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.*;
public class AndroidIntentGolfApp extends Activity {
        TextView label1;
        EditText text1;
        Button btnCallActivity2;
```

Example 2: Intents

Example2. ***Calling a sub-activity, receiving results.***

```
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      try{
       setContentView(R.layout.main);
       label1= (TextView)findViewById(R.id.label1);
       text1= (EditText)findViewById(R.id.text1);
       btnCallActivity2=
(Button)findViewById(R.id.btnPickContact);
       btnCallActivity2.setOnClickListener(new ClickHandler());
      } catch(Exception e) {
       Toast.makeText(getBaseContext(),
       e.getMessage(), Toast.LENGTH_LONG).show();
      }
} //onCreate
```

Example 2: Intents

```
private class ClickHandler implements OnClickListener{
    @Override
    public void onClick(View v) {
        try{
                // myDatarefer to: content://contacts/people/
                String myData= text1.getText().toString();
                //you may also try ACTION_VIEW instead
                Intent myActivity2 = new
    Intent(Intent.ACTION_PICK,
                            Uri.parse(myData));
                // start myActivity2.
                // Tell it that our requestCodeID(or nickname) is
    222
                startActivityForResult(myActivity2, 222);
                // Toast.makeText(getApplicationContext(),
                // "I can't wait for you", 1).show();
        } catch(Exception e) {
                label1.setText(e.getMessage());
        }
    }//onClick
}//ClickHandler
```

When this activity exists, onActivityResult() method will be called with the given requestCode.

Example 2: Intents

```
@Override
protected void onActivityResult(int requestCode,  int resultCode,
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        try{
        // use requestCodeto find out who is talking back to us
        switch(requestCode){
        case(222): {
              // 222 is our friendly contact-picker activity
              if (resultCode== Activity.RESULT_OK) {
                    String selectedContact= data.getDataString();
                    // it will return an URI that looks like:
                    // content://contacts/people/n
                    // where n is the selected contacts' ID
                    label1.setText(selectedContact.toString());

                    //show a 'nice' screen with the selected contact
                    Intent myAct3 = new Intent (Intent.ACTION_VIEW,
                              Uri.parse(selectedContact));
                    startActivity(myAct3);
              }
```

Example 2: Intents

```
              else {
                        //user pressed the BACK button
                        label1.setText("Selection CANCELLED " + requestCode+ " "+
resultCode);
              }
              break;
            }
    }  //switch
  }
    catch(Exception e) {
    Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_LONG).show();
    }
  } // onActivityResult
} //IntentDemo2
```

Example 2: Intents

Example2. *Calling a sub-activity, receiving results.*

```
<?xml version="1.0" encoding="utf-8"?>
 <LinearLayoutxmlns:android="http://schemas.android.com/apk/res/andr
oid"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent" >
      <TextView  android:id="@+id/label1"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:background="#ff0000cc"
          android:text="This is Activity1"
          android:textStyle="bold"
          android:textSize="20sp"/>
      <EditText  android:id="@+id/text1"
          android:layout_width="fill_parent"
          android:layout_height="54px"
          android:text="content://contacts/people/"
          android:textSize="18sp" />
      <Button android:id="@+id/btnPickContact"
          android:layout_width="149px"
          android:layout_height="wrap_content"
          android:text="Pick a Contact"
          android:textStyle="bold" />
</LinearLayout>
```