Intent & Service

Intents

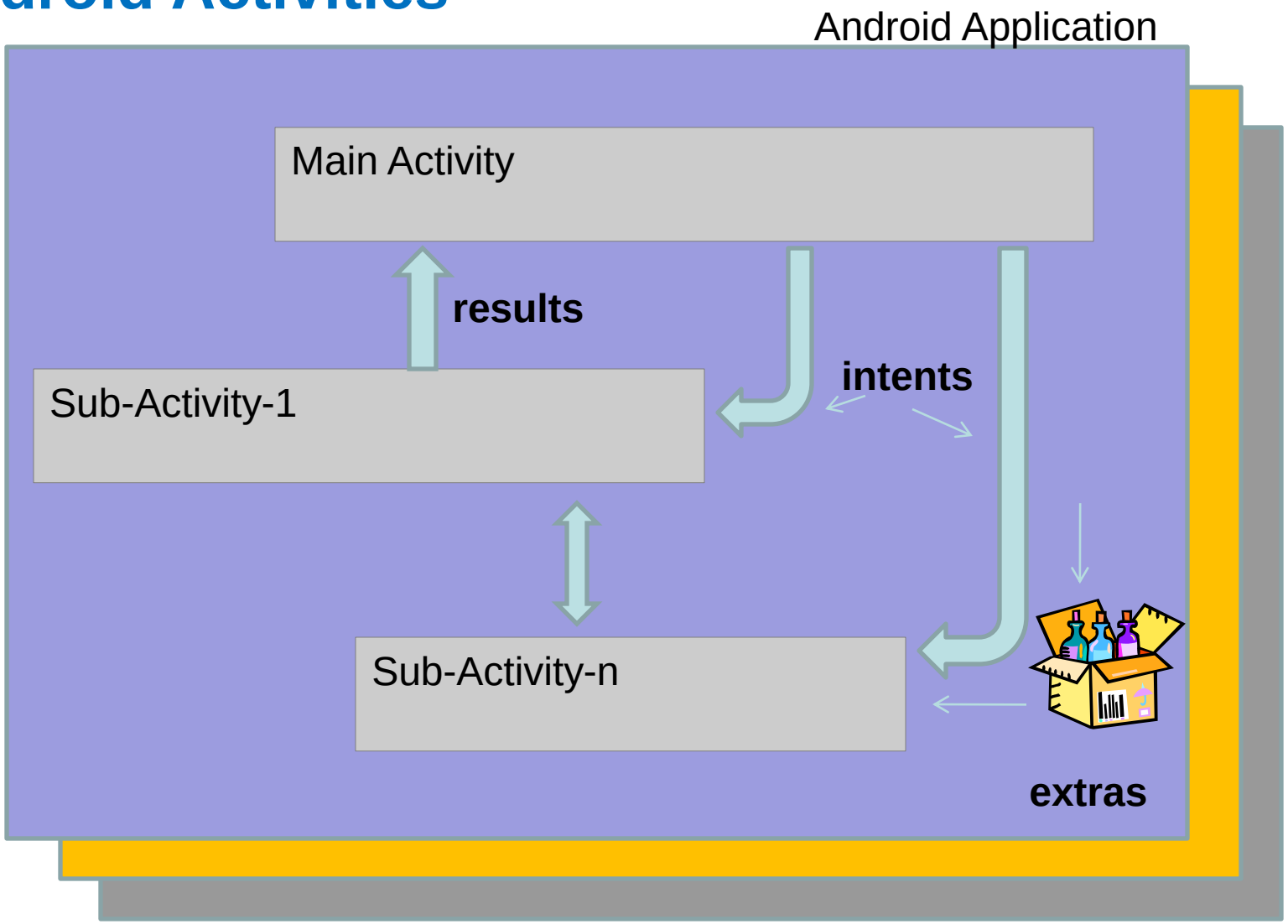
Android Activities

Một chương trình android có thể có nhiều activity.

- Một activity dùng phương thức *setContentView*(...) để thiết lập giao diện tương tác người dùng.
- Các activity độc lập, nhưng chúng có thể cộng tác với nhau nhằm hoàn thành 1 chức năng nào đó, và giao tiếp với nhau.
- Điển hình mỗi ứng dụng có 1 activity là chính (main) activity này được gọi khi chương trình khởi tạo.
- Khi một activity này chuyển sang một activity khác thì yêu cầu phải thực hiện một ***intent***.
- Các activity tương tác với nhau thông qua chế độ bất đồng bộ.

Intents

Android Activities



Intents

Taken from: http://code.google.com/android/reference/android/content/Intent.html

Intents are invoked using the following options

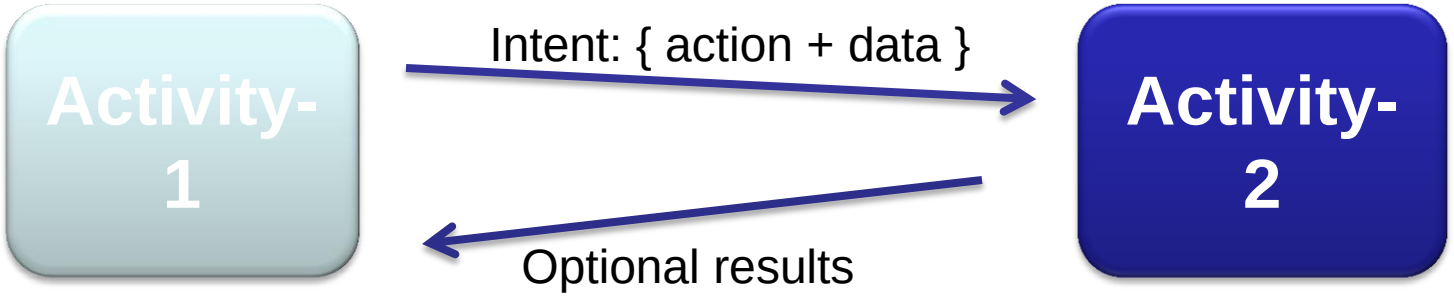
startActivity **(intent)**	Gọi 1 activity
sendBroadcast **(intent)**	Gửi đến các *BroadcastReceiver* liên quan
startService**(intent)** or ***bindService*****(intent, …)**	Giao tiếp với các dịch vụ nền.

Intents

Taken from: http://code.google.com/android/reference/android/content/Intent.html

2 tham số chính của Intent:

1. **Action** một số được xây dựng sẵn, như là `ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, ...`
   hoặc *user-created-activity*
2. **Data** Dữ liệu cho các hoạt động như là số điện thoại để gọi (được biểu diễn như là **Uri**).



Intents

Taken from: http://code.google.com/android/reference/android/content/Intent.html

Ví dụ 1 intent được sử dụng như sau:

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

startActivity (myActivity);
```

Built-in
or
user-created
activity

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

Intents

Taken from: http://code.google.com/android/reference/android/content/Intent.html

Ví dụ các cặp action/data :

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 2-*nd* person.

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

Danh sách các action được android xây dựng sẵn (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	

For a list of actions see:
http://developer.android.com/reference/android/content/Intent.html

Intents

ACTION_AIRPLANE_MODE_CHANGED

ACTION_ALL_APPS

ACTION_ANSWER

ACTION_ATTACH_DATA

ACTION_BATTERY_CHANGED

ACTION_BATTERY_LOW

ACTION_BATTERY_OKAY

ACTION_BOOT_COMPLETED

ACTION_BUG_REPORT

ACTION_CALL

ACTION_CALL_BUTTON

ACTION_CAMERA_BUTTON

ACTION_CHOOSER

ACTION_CLOSE_SYSTEM_DIALOGS

ACTION_CONFIGURATION_CHANGED

ACTION_CREATE_SHORTCUT

ACTION_DATE_CHANGED

ACTION_DEFAULT

ACTION_DELETE

ACTION_DEVICE_STORAGE_LOW

ACTION_DEVICE_STORAGE_OK

ACTION_DIAL

ACTION_DOCK_EVENT

ACTION_EDIT

ACTION_EXTERNAL_APPLICATIONS_AVAILABLE

ACTION_EXTERNAL_APPLICATIONS_UNAVAILABLE

ACTION_FACTORY_TEST

ACTION_GET_CONTENT

ACTION_GTALK_SERVICE_CONNECTED

ACTION_GTALK_SERVICE_DISCONNECTED

ACTION_HEADSET_PLUG

ACTION_INPUT_METHOD_CHANGED

ACTION_INSERT

ACTION_INSERT_OR_EDIT

ACTION_LOCALE_CHANGED

ACTION_MAIN

ACTION_MANAGE_PACKAGE_STORAGE

ACTION_MEDIA_BAD_REMOVAL

ACTION_MEDIA_BUTTON

ACTION_MEDIA_CHECKING

ACTION_MEDIA_EJECT

ACTION_MEDIA_MOUNTED

ACTION_MEDIA_NOFS

ACTION_MEDIA_REMOVED

ACTION_MEDIA_SCANNER_FINISHED

ACTION_MEDIA_SCANNER_SCAN_FILE

ACTION_MEDIA_SCANNER_STARTED

ACTION_MEDIA_SHARED

ACTION_MEDIA_UNMOUNTABLE

ACTION_MEDIA_UNMOUNTED

ACTION_MY_PACKAGE_REPLACED

ACTION_NEW_OUTGOING_CALL

ACTION_PACKAGE_ADDED

ACTION_PACKAGE_CHANGED

ACTION_PACKAGE_DATA_CLEARED

ACTION_PACKAGE_FIRST_LAUNCH

ACTION_PACKAGE_INSTALL

ACTION_PACKAGE_REMOVED

ACTION_PACKAGE_REPLACED

ACTION_PACKAGE_RESTARTED

ACTION_PASTE

ACTION_PICK

ACTION_PICK_ACTIVITY

ACTION_POWER_CONNECTED

ACTION_POWER_DISCONNECTED

ACTION_POWER_USAGE_SUMMARY

ACTION_PROVIDER_CHANGED

ACTION_REBOOT

ACTION_RUN

ACTION_SCREEN_OFF

ACTION_SCREEN_ON

ACTION_SEARCH

ACTION_SEARCH_LONG_PRESS

ACTION_SEND

ACTION_SENDTO

ACTION_SEND_MULTIPLE

ACTION_SET_WALLPAPER

ACTION_SHUTDOWN

ACTION_SYNC

ACTION_SYSTEM_TUTORIAL

ACTION_TIMEZONE_CHANGED

ACTION_TIME_CHANGED

ACTION_TIME_TICK

ACTION_UID_REMOVED

ACTION_UMS_CONNECTED

ACTION_UMS_DISCONNECTED

ACTION_USER_PRESENT

ACTION_VIEW

ACTION_VOICE_COMMAND

ACTION_WALLPAPER_CHANGED

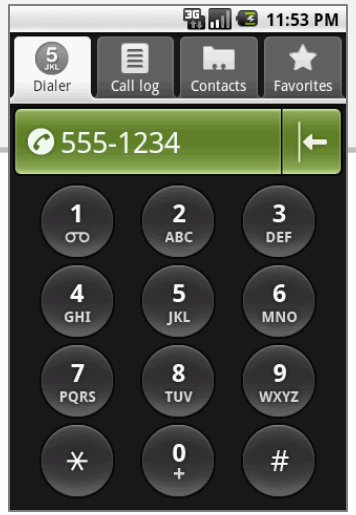
ACTION_WEB_SEARCH

Intents

Taken from: http://code.google.com/android/reference/android/content/Intent.html

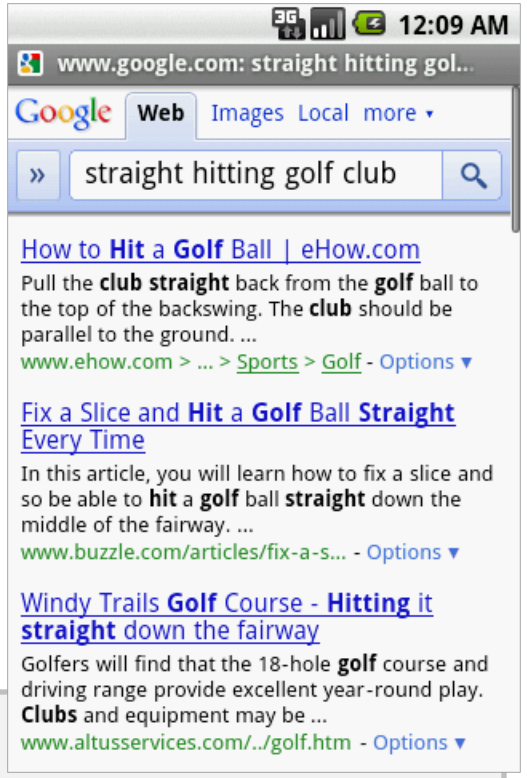
Ví dụ : gọi intent thực hiện gọi điện thoại

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



Intents

Taken from: http://code.google.com/android/reference/android/content/Intent.html



Intents - Secondary Attributes

Ngoài thuộc tính action/ data thì intent còn 1 thuộc tính là:

1. Category
2. Components
3. Type
4. Extras

Example: sử dụng chức năng tìm kiếm của google

```
Intent intent = new Intent (Intent.ACTION_WEB_SEARCH );

intent.putExtra(SearchManager.QUERY,
                "straight hitting golf clubs");

startActivity(intent);
```

Secondary data

Apparently the Google answer is 'none'

Intents

Taken from: http://code.google.com/android/reference/android/content/Intent.html

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", "are we playing golf next Saturday?");

startActivity(intent);
```

"address", "sms_body" are keywords

Intents

Taken from: http://code.google.com/android/reference/android/content/Intent.html

Intents - Secondary Attributes

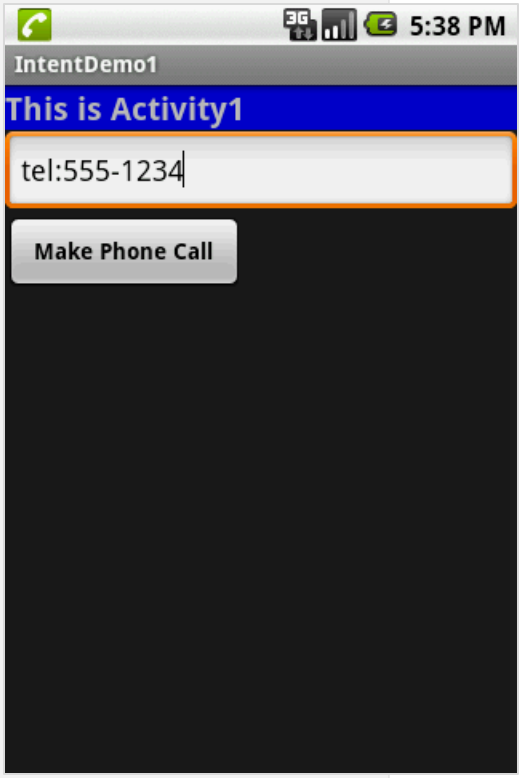
Example: Showing Pictures (using extra attributes)

```
Intent myIntent = new Intent();

myIntent.setType("image/pictures/*");
myIntent.setAction(Intent.ACTION_GET_CONTENT);

startActivity(myIntent);
```

Intents

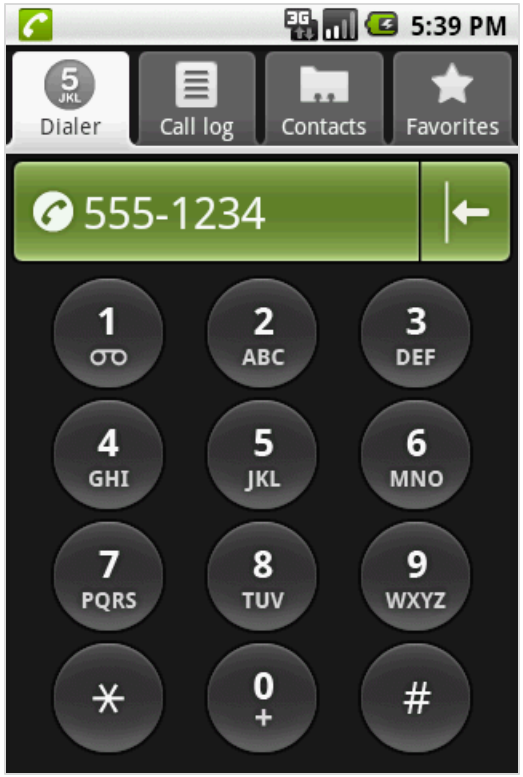
1. Ví dụ : 1 activity cho phép người dùng nhập số gọi, sau đó gọi activity thực hiện cuộc gọi (Activity này có sẵn trong hệ thống).

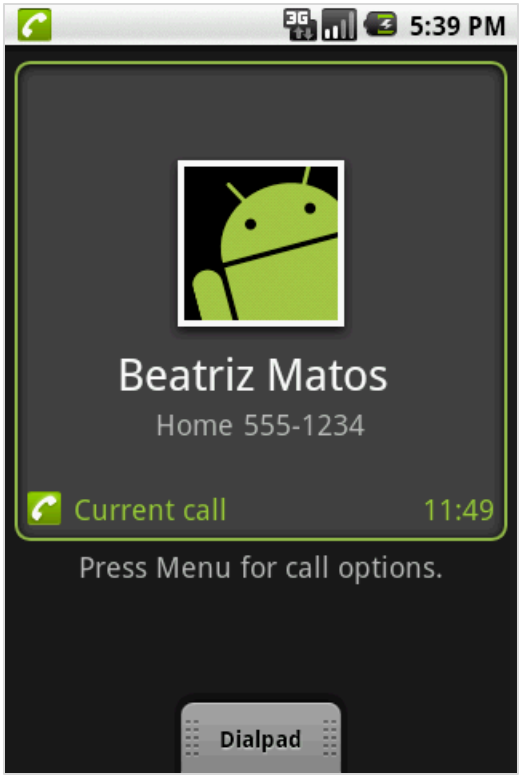
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"  >
<TextView
      android:id="@+id/label1"
      android:layout_width="fill_parent"
      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:555-1234"
      android:textSize="18sp"  />

<Button
      android:id="@+id/btnCallActivity2"
      android:layout_width="149px"
      android:layout_height="wrap_content"
      android:text="Make Phone Call"
      android:textStyle="bold" />
</LinearLayout>
```

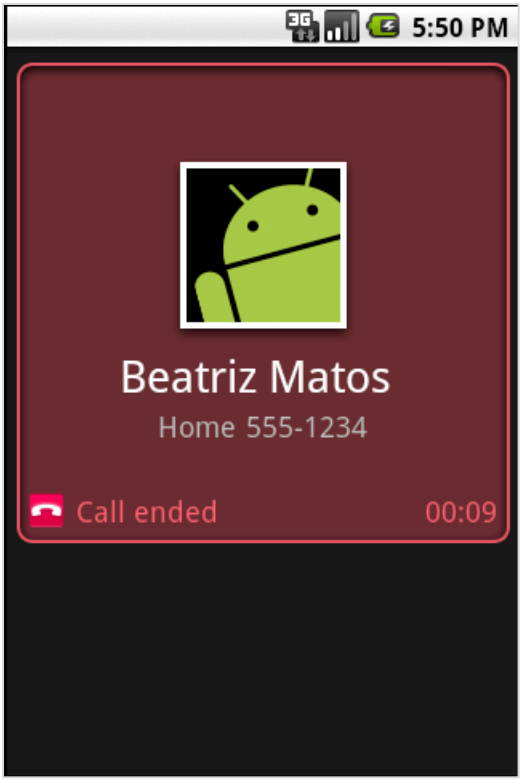


Intents

1. Ví dụ : 1 activity cho phép người dùng nhập số gọi, sau đó gọi activity thực hiện cuộc gọi (Activity này có sẵn trong hệ thống).







Intents

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

```
//IntentDemo1_Intent: making a phone call
package cis493.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 IntentDemo1 extends Activity {
    TextView label1;
    EditText text1;
    Button   btnCallActivity2;
```

Intents

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

```
@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);
        btnCallActivity2.setOnClickListener(new ClickHandler());
    }
   catch (Exception e) {
        Toast.makeText(getBaseContext(), e.getMessage(),
                       Toast.LENGTH_LONG).show();
    }
  }//onCreate
```


18

Intents

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

```
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:555-1234"
            // 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
}//IntentDemo1
```

Intents

1. A Complete Example: Activity1 displays an interface that accepts from the user 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="cis493.intents"
      android:versionCode="1"
      android:versionName="1.0">
<application android:icon="@drawable/icon"
             android:label="@string/app_name">
  <activity  android:name=".IntentDemo1"
             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="3" />

</manifest>
```

Action/category

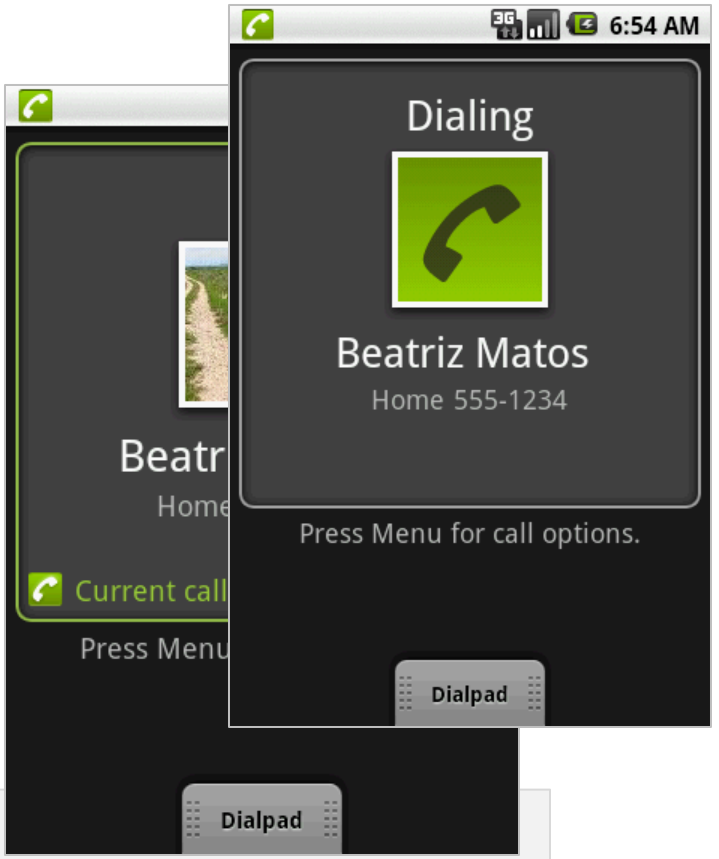
Intents

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

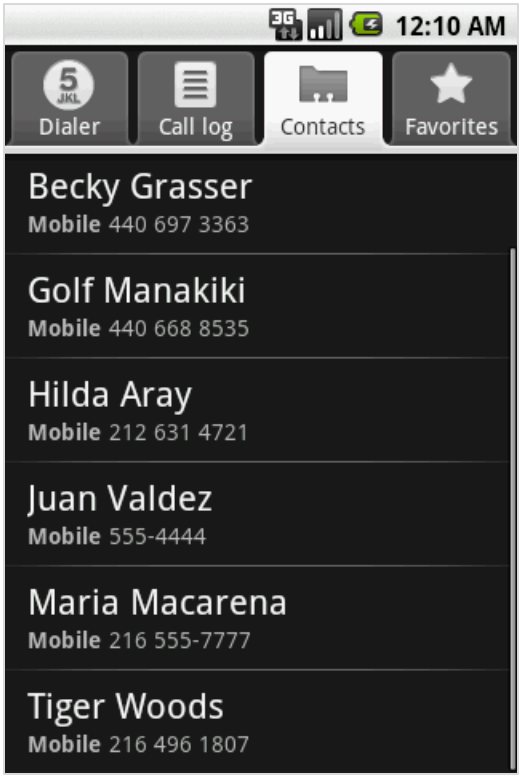
```
String myData = "tel:555-1234";

Intent myActivity2 = new Intent(Intent.ACTION_CALL,
                                Uri.parse(myData));

startActivity(myActivity2);
```

Needs Permission:

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


22

Intents



More Examples: Using Standard Actions

Show all your Contacts

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

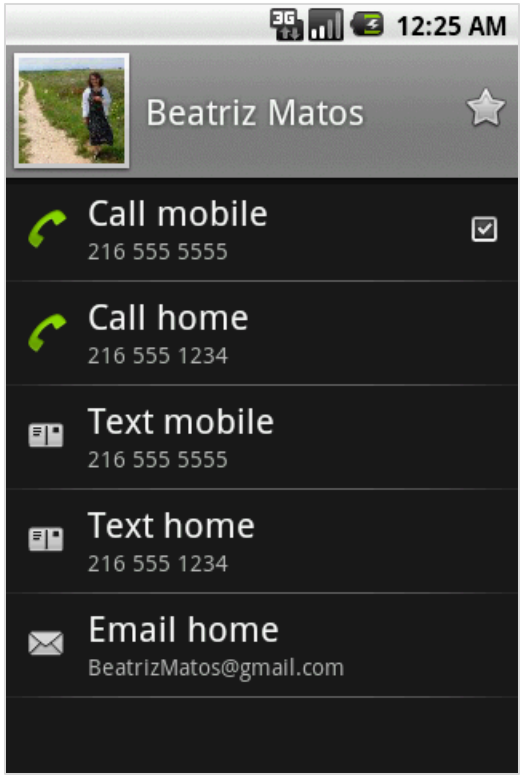
```
String myData = "content://contacts/people/";

Intent myActivity2 = new Intent(Intent.ACTION_VIEW,
                                Uri.parse(myData));

startActivity(myActivity2);
```

Intents

More Examples: Using Standard Actions

Show a Particular Contact (ID = 210)

Modify the *complete* example1 replacing the method ‘ClickHandler’ with the following code



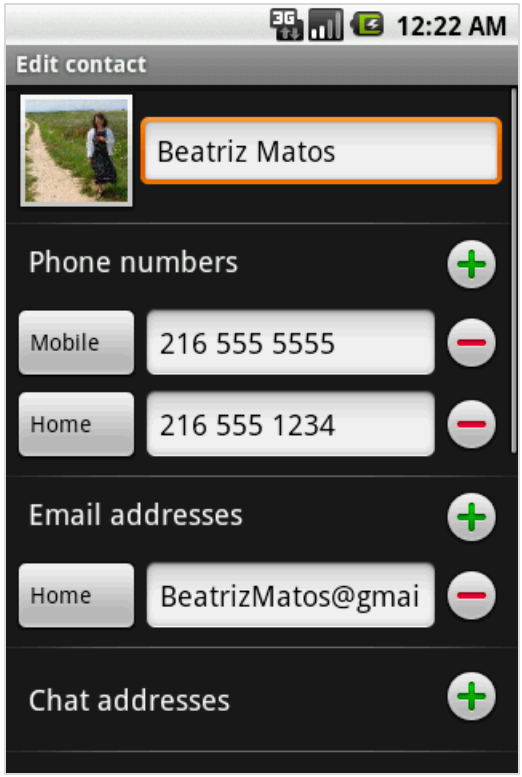

```
String myData = ContactsContract.Contacts.CONTENT_URI + "/" + "210";

Intent myActivity2 = new Intent(Intent.ACTION_VIEW,
                                Uri.parse(myData));

startActivity(myActivity2);
```

Intents

More Examples: Using Standard Actions

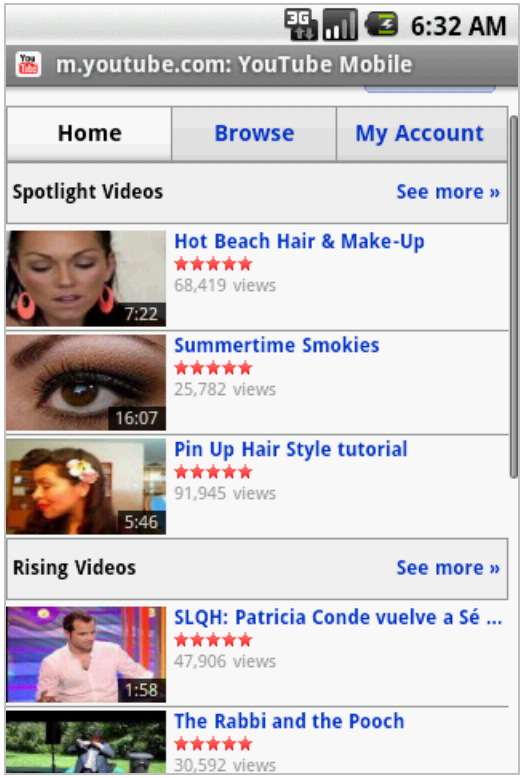
Edit a Particular Contact (ID = 210)

Modify the *complete* example1 replacing the method ‘ClickHandler’ with the following code




```
String myData = ContactsContract.Contacts.CONTENT_URI + "/" + "210";

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:

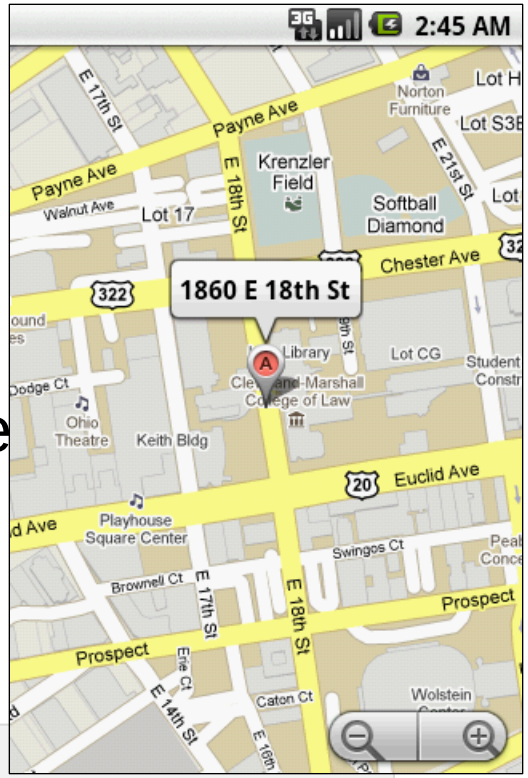
```
<uses-permission android:name="android.permission.INTERNET" />
```

Intents

More Examples: Using Standard Actions

Geo Mapping an Address

Provide a geoCode expression holding a stre
address (or place, such as ‘golden gate ca’)
Replace spaces with ‘+’.




```
String geoCode =
        "geo:0,0?q=1860+east+18th+street+cleveland+oh";
Intent intent = new Intent(Intent.ACTION_VIEW,
                           Uri.parse(geoCode));
startActivity(intent);
```

Modify the Manifest adding the following requests:

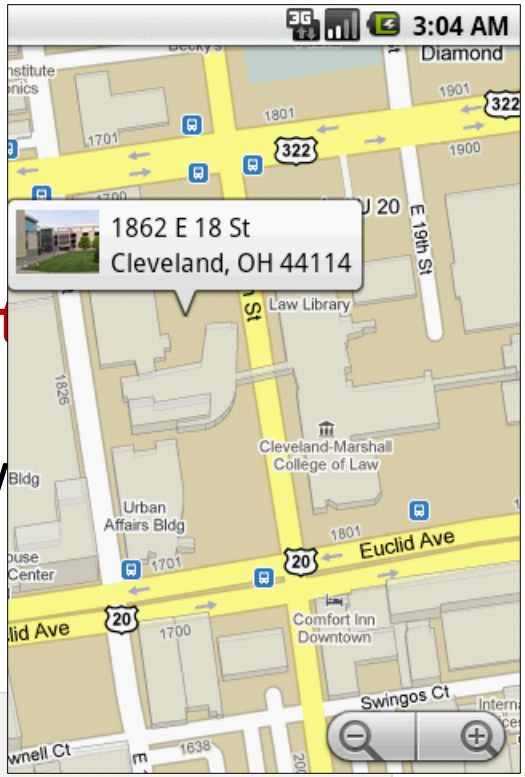
```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
```

Intents

More Examples: Using Standard Actions

Geo Mapping Coordinates (latitude, longit

Provide a geoCode holding latitude and longitude (also an addittional zoom '**?z=xx**' w xx in range 1..23)




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

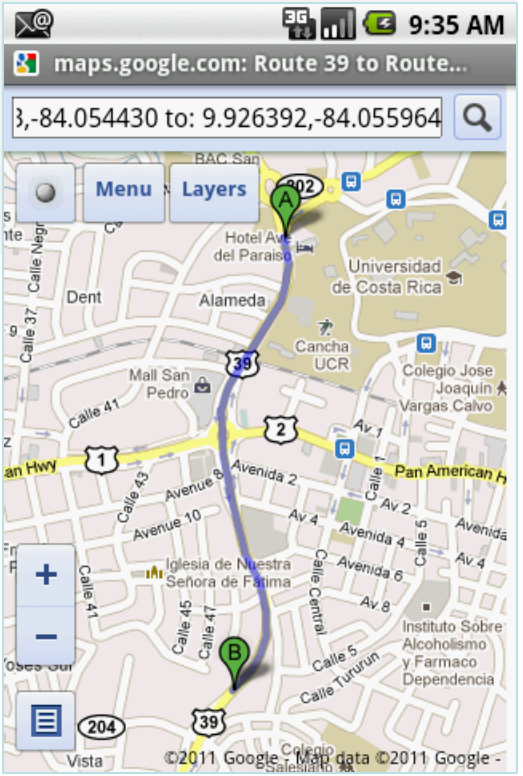
Modify the Manifest adding the following requests:

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

Intents

More Examples

```
// Getting driving directions: how to go from loaction A to location B?

String sourceDestination =
   "http://maps.google.com/maps?saddr=9.938083,-84.054430&daddr=9.926392,-84.055964";

Intent intent = new Intent(
                  android.content.Intent.ACTION_VIEW,
                  Uri.parse(sourceDestination));

startActivity(intent);
```

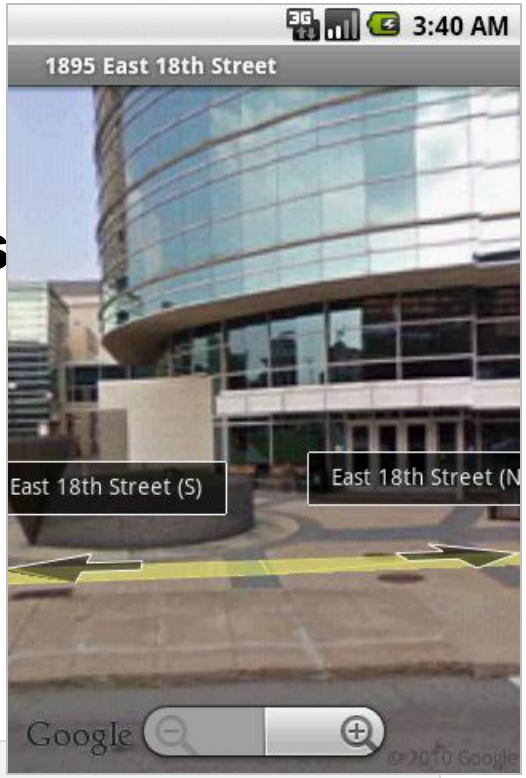


Intents

More Examples

```
// use a mnemonic  to articulate an address
String thePlace = "Cleveland State University";

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

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

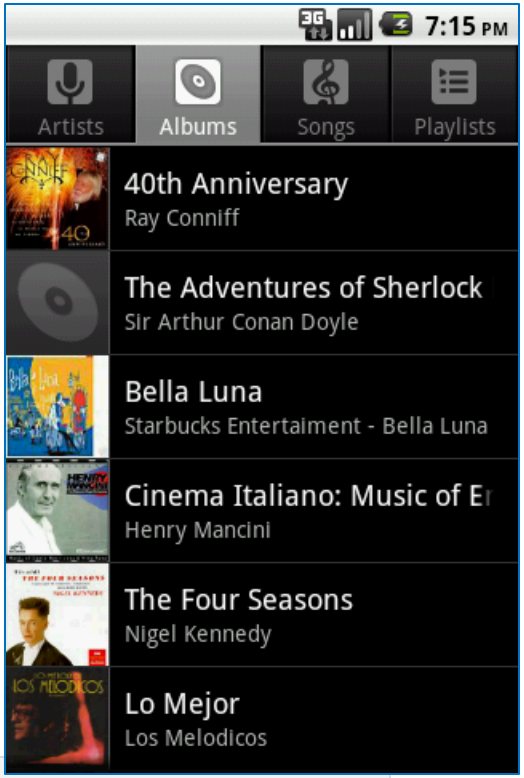
Reference: http://developer.android.com/guide/appendix/g-app-intents.html

```
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



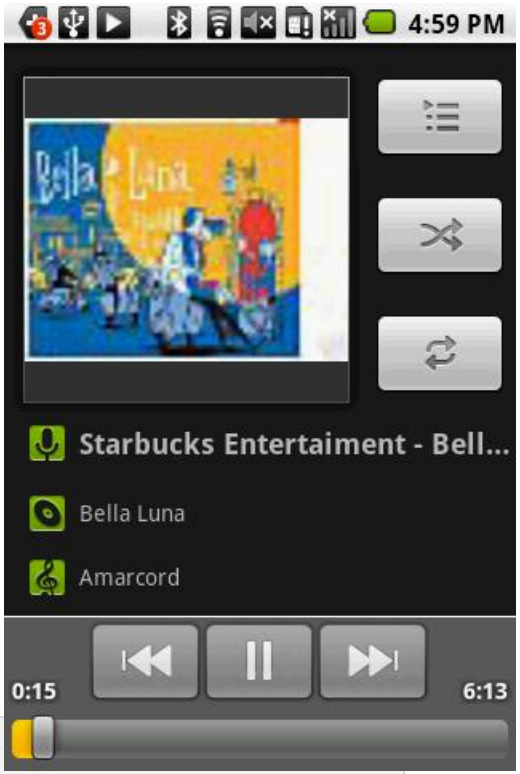
More Examples: Using Standard Actions

Launching the Music Player

Reference: http://developer.android.com/guide/appendix/g-app-intents.html

```
//launch music player

Intent myActivity2 =
       new Intent("android.intent.action.MUSIC_PLAYER");

startActivity(myActivity2);
```

Intents



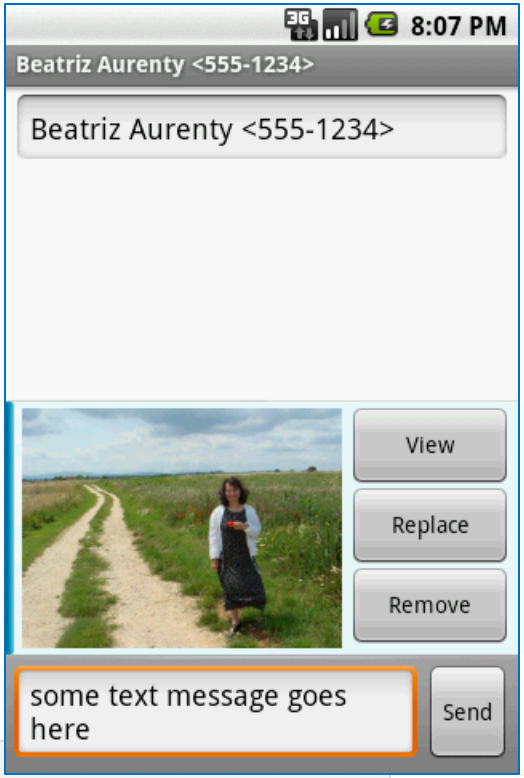
More Examples: Using Standard Actions

Playing a song stored in the SD card

Reference: http://developer.android.com/guide/appendix/g-app-intents.html

```
// 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

Reference: http://developer.android.com/guide/appendix/g-app-intents.html

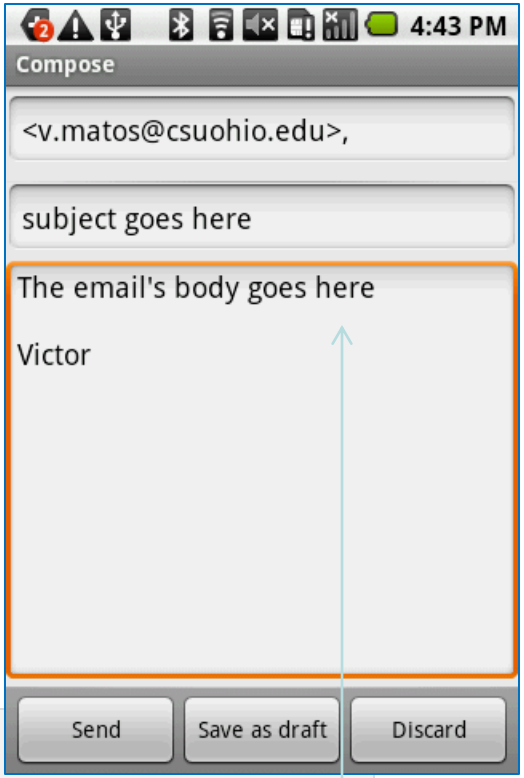
```
//send mms attach picture #1 to it

Uri uri = Uri.parse("content://media/external/images/media/1");

Intent 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

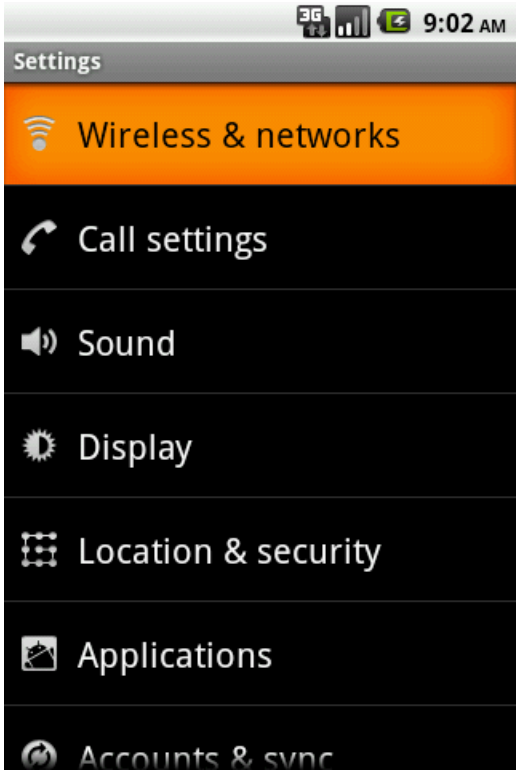
Sending Email

Reference: http://developer.android.com/guide/appendix/g-app-intents.html




```
// send email
Uri uri = Uri.parse("mailto:v.matos@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



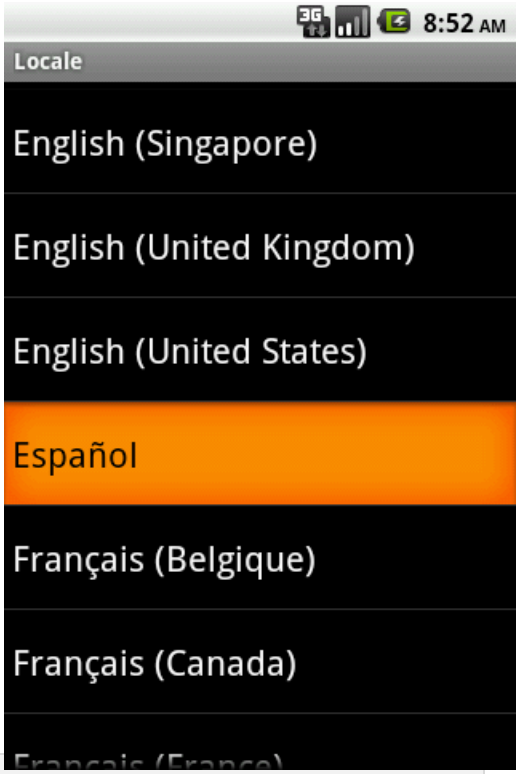
More Examples: Using Standard Actions

Setting System

Reference: http://developer.android.com/reference/android/provider/Settings.html

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

startActivity(intent);
```

Intents



More Examples: Using Standard Actions

Setting System Locale: Language & Keyboard

Reference: http://developer.android.com/reference/android/provider/Settings.html

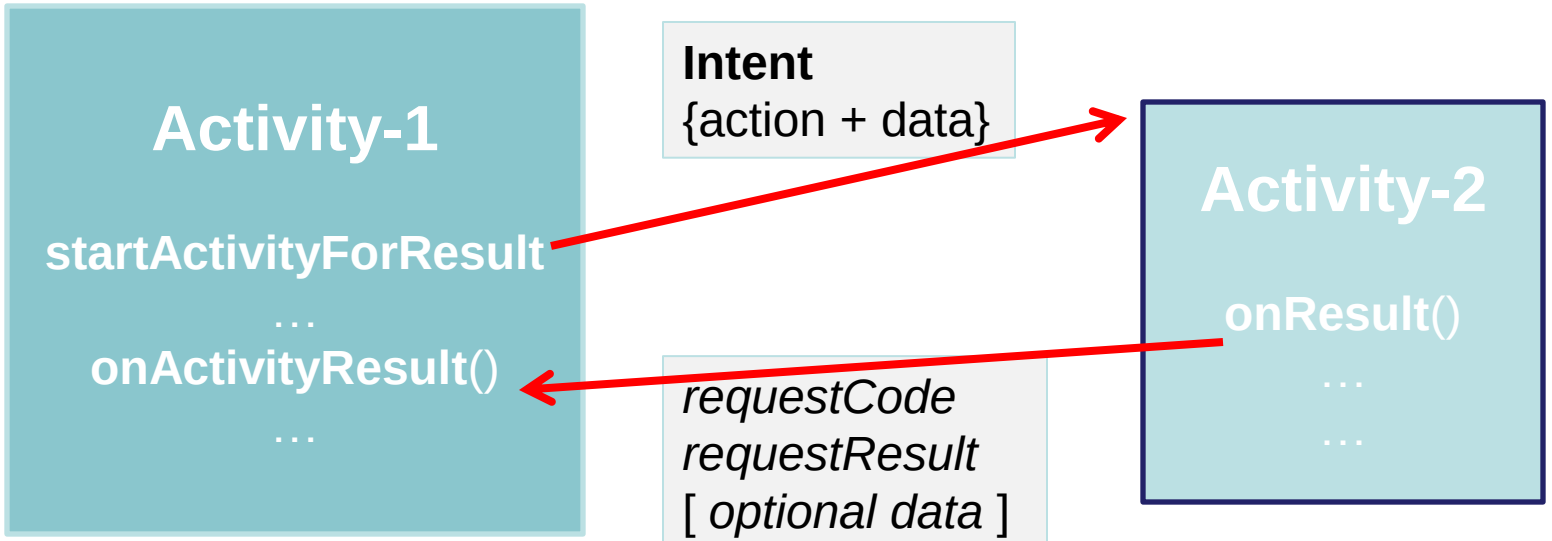
```
Intent intent = new Intent(
            android.provider.Settings.ACTION_LOCALE_SETTINGS);
startActivity(intent);
```

Intents

Android Intents

An *activity* usually presents a single visual user interface from which a number of actions could be performed.

Moving from one activity to another is accomplished by having the current activity start the next one through so called ***intents***.



39

Intents

Android Bundles

Most programming languages support the notion of **IPC** *method-calling* with arguments flowing birectionally from the caller to the invoked method.

In android the calling activity issues an invocation to another activity using an **Intent** object.

Notably in Android, *the caller does not stop waiting* for the called activity to return results. Instead a listening-method [*onActivityResult(...)*] should be used.

Intents

Android Bundles

Normally the IPC expressions *actual parameter list*, and *formal parameter list* are used to designated the signature of particpating arguments, and the currently supplied data.

Instead of using the traditional *formal / actual parameter lists,* Android relies on the concept of Intents to establish Inter-process-communication.

Intents optionally carry a named actual list or **bundle** for data exchange.

41

Intents

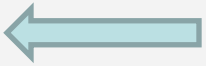
Android Bundles

The Android **Bundle** container is a simple mechanism used to pass data between activities.

A **Bundle** is a type-safe collection of **<name, value>** pairs.

There is a set of ***putXXX*** and ***getXXX*** methods to store and retrieve (single and array) values of primitive data types from/to the bundles. For example

```
Bundle myBundle = new Bundle();
myBundle.putDouble ("var1", 3.1415);
...
Double v1 = myBundle.getDouble("var1");
```



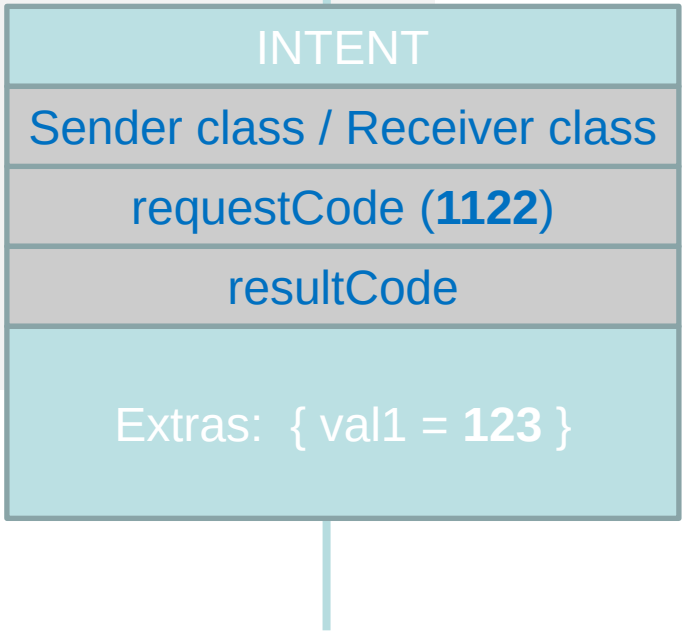
42

Intents

Android Intents & Bundles

Activity1: Sender → **Activity2: Receiver**

```
Intent myIntentA1A2 = new Intent  (Activity1.this,
                                   Activity2.class);

Bundle myBundle1 = new Bundle();
myBundle1.putInt ("val1", 123);

myIntentA1A2.putExtras(myBundle1);

startActivityForResult(myIntentA1A2, 1122);
```



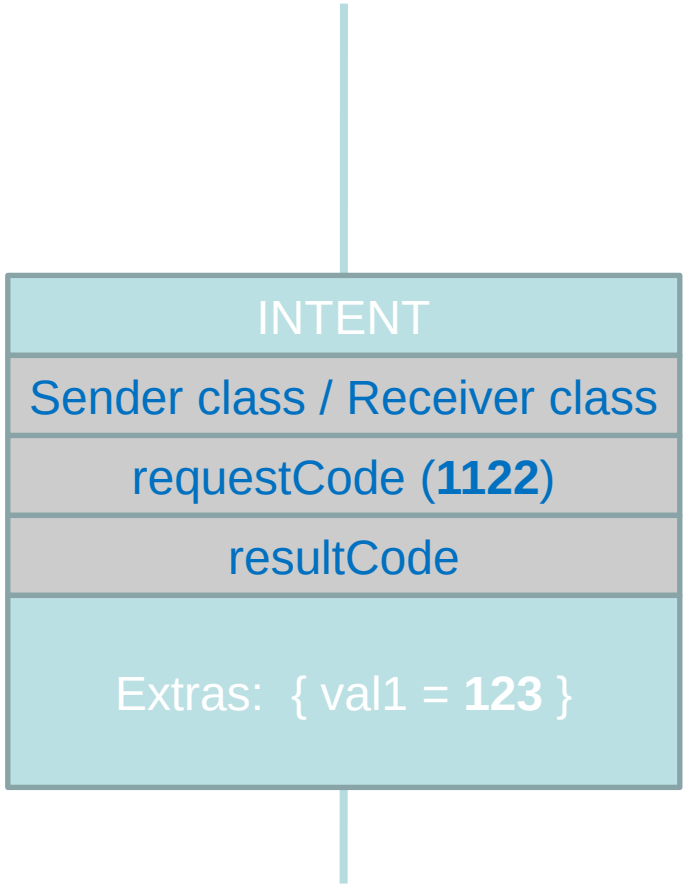
43

Intents

Android Intents & Bundles

Activity1: Sender

Activity2: Receiver




```
Intent myCallerIntent2 = getIntent();

Bundle myBundle = myCallerIntent.getExtras();

int val1 = myBundle.getInt("val1");
```

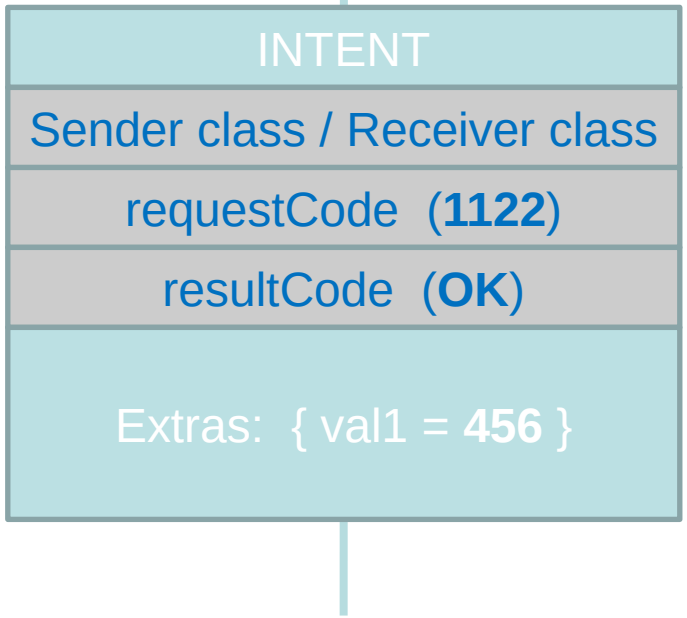
44

Intents

Android Intents & Bundles

Activity1: Sender ← **Activity2: Receiver**

```
myBundle.putString("val1", 456 );

myCallerIntent.putExtras(myBundle);

setResult(Activity.RESULT_OK,
          myCallerIntent);
```



45

Intents

Android Bundles

Available at:

http://developer.android.com/reference/android/os/Bundle.html

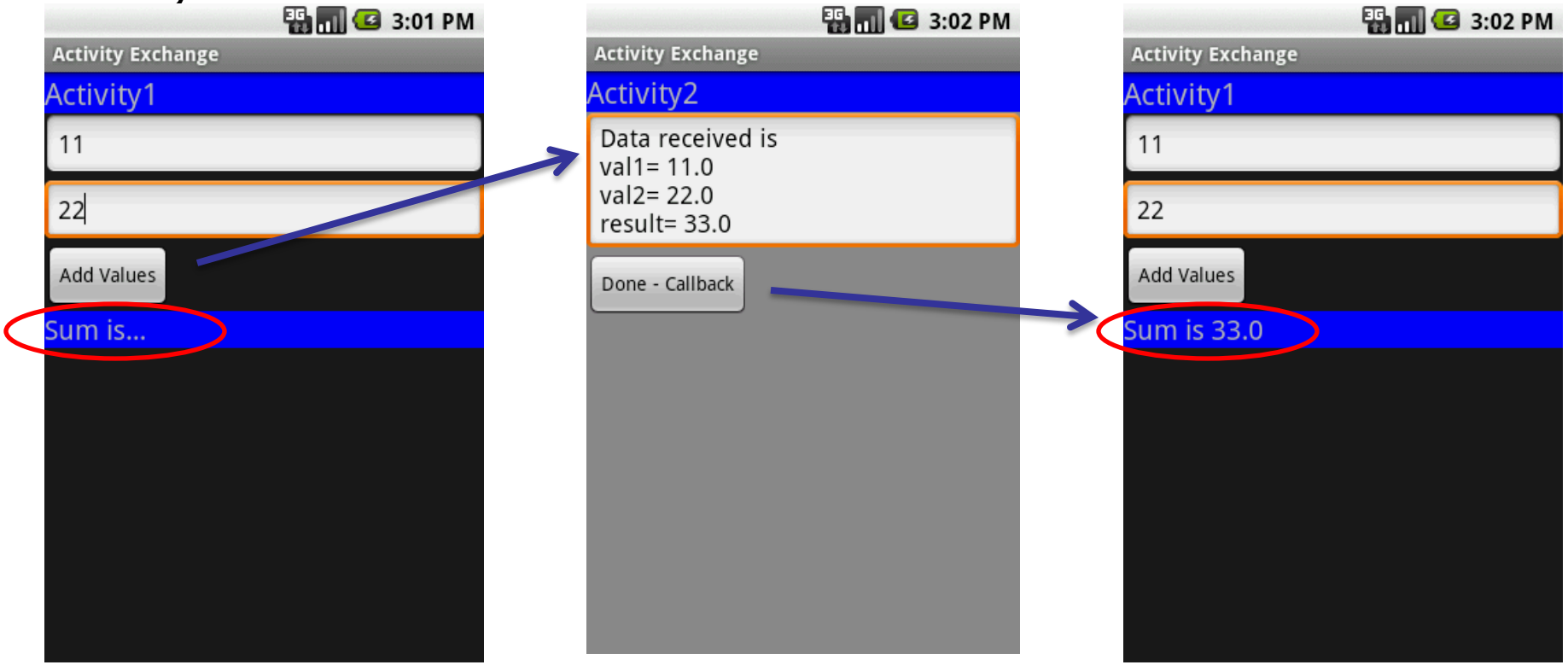
Example of Public Methods

void	clear() Removes all elements from the mapping of this Bundle.
Object	clone() Clones the current Bundle.
boolean	containsKey(String key) Returns true if the given key is contained in the mapping of this Bundle.
void	putIntArray(String key, int[] value) Inserts an int array value into the mapping of this Bundle, replacing any existing value for the given key.
void	putString(String key, String value) Inserts a String value into the mapping of this Bundle, replacing any existing value for the given key.
void	putStringArray(String key, String[] value) Inserts a String array value into the mapping of this Bundle, replacing any existing value for the given key.
void	putStringArrayList(String key, ArrayList<String> value) Inserts an ArrayList value into the mapping of this Bundle, replacing any existing value for the given key.
void	remove(String key) Removes any entry with the given key from the mapping of this Bundle.
int	size() Returns the number of mappings contained in this Bundle.

Intents

Tutorial 1. Activity Excahange

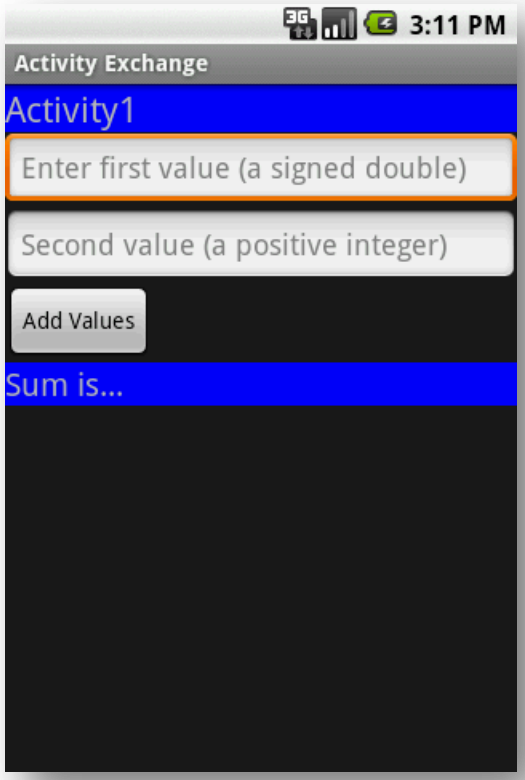
Activity1 collects two values from its UI and calls Activity2 to compute the sum of them. The result is sent back from Activity 2 to Activity1.



Intents

Tutorial 1. Activity Excahange

Step1. Create GUI for Activity1(main1.xml



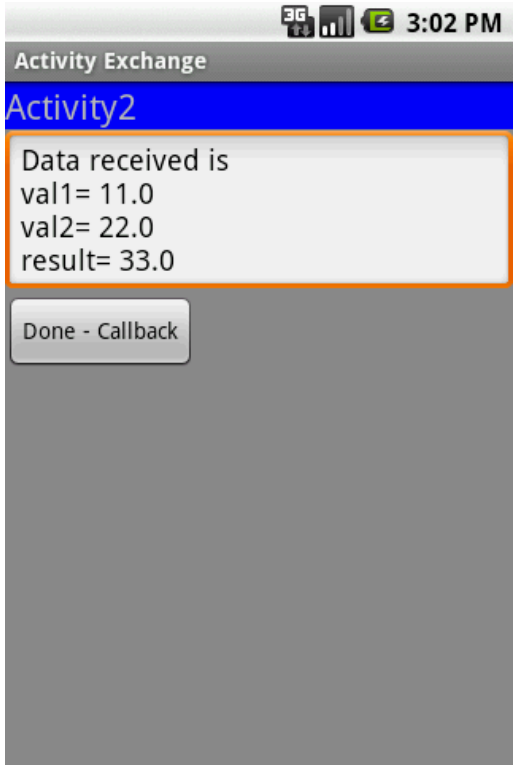

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"   android:layout_width="fill_parent"
android:layout_height="fill_parent"  >
<TextView
        android:text="Activity1"
        android:textSize="22sp"
        android:background="#ff0000ff"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
<EditText
        android:hint="Enter first value (a signed double)"
        android:id="@+id/EditText01"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal|numberSigned|number" />
<EditText
        android:hint="Second value (a positive integer)"
        android:id="@+id/EditText02"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="number" />
<Button
        android:text="Add Values"
        android:id="@+id/btnAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
<TextView
        android:background="#ff0000ff"
        android:text="Sum is..."
        android:textSize="28sp"
        android:id="@+id/TextView01"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Note. The element **android:inputStyle** indicates the first value could be numeric, with optional decimals and sign.

Intents

Tutorial 1. Activity Excahange

Step2. Create GUI for Activity2(main2.xml)



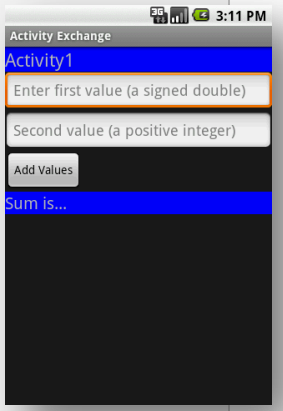

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="#ff888888">

<TextView
        android:text="Activity2"
        android:textSize="22sp"
        android:background="#ff0000ff"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

<EditText
        android:text="Data reveived..."
        android:id="@+id/etDataReceived"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

<Button
        android:text="Done - Callback"
        android:id="@+id/btnDone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Intents

Tutorial 1. Activity Excahange

Step3. Activity1. After clicking the button data, from UI is put in a bundle and sent to Activity2. A listener remains alert waiting for results to come from the called

```
package cis493.matos.intent2b;
// Activity1: get two input values from user, put them in a bumble. call Activity2 to add the two numbers, show result
import ...;

public class Activity1 extends Activity {
    EditText txtVal1;
    EditText txtVal2;
    TextView lblResult;
    Button   btnAdd;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main1);
        txtVal1 = (EditText)findViewById(R.id.EditText01);
        txtVal2 = (EditText)findViewById(R.id.EditText02);
        lblResult = (TextView) findViewById(R.id.TextView01);
        btnAdd = (Button) findViewById(R.id.btnAdd);
        btnAdd.setOnClickListener(new OnClickListener() {
          @Override
          public void onClick(View v) {
             // get values from the UI
             Double v1 = Double.parseDouble(txtVal1.getText().toString());
             Double v2 = Double.parseDouble(txtVal2.getText().toString());

             // create intent to call Activity2
             Intent myIntentA1A2 = new Intent (Activity1.this, Activity2.class);
             // create a container to ship data
             Bundle myData = new Bundle();

             // add <key,value> data items to the container
             myData.putDouble("val1", v1);
             myData.putDouble("val2", v2);

             // attach the container to the intent
             myIntentA1A2.putExtras(myData);

             // call Activity2, tell your local listener to wait for response
             startActivityForResult(myIntentA1A2, 101);
          }
       });
    }//onCreate
```

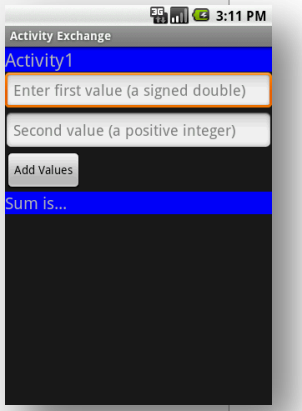


Intents

Tutorial 1. Activity Excahange *cont.*

Step3. Activity1. After clicking the button data, from UI is put in a bundle and sent to Activity2. A listener remains alert waiting for results to come from the called

```
    ////////////////////////////////////////////////////////////////////////
    // local listener receiving callbacks from other activities

   @Override
   protected void onActivityResult(int requestCode, int resultCode, Intent data) {
      super.onActivityResult(requestCode, resultCode, data);

      try   {
         if ((requestCode == 101 ) && (resultCode == Activity.RESULT_OK)){

            Bundle myResults = data.getExtras();

            Double vresult = myResults.getDouble("vresult");

            lblResult.setText("Sum is " + vresult);

         }
      }
      catch (Exception e) {
         lblResult.setText("Problems - " + requestCode + " " + resultCode);
      }
   }//onActivityResult

}//Activity1
```

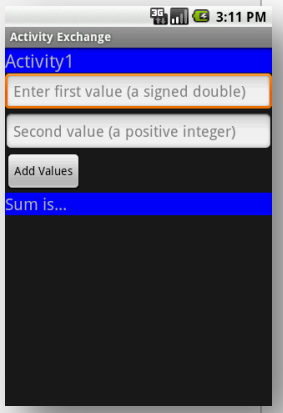


Intents

Tutorial 1. Activity Excahange *cont.*

Step4. Activity2. Called from Activity1. Extracts input data from the bundle attached to the intent. Performs local computation. Adds result to bundle. Returns

```
package cis493.matos.intent2b;

import . . .;

public class Activity2 extends Activity implements OnClickListener{
    EditText dataReceived;
    Button  btnDone;

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

        setContentView(R.layout.main2);
        dataReceived = (EditText) findViewById(R.id.etDataReceived);
        btnDone = (Button) findViewById(R.id.btnDone);
        btnDone.setOnClickListener(this);

        // pick call made to Activity2 via Intent
        Intent myLocalIntent = getIntent();

        // look into the bundle sent to Activity2 for data items
        Bundle myBundle =  myLocalIntent.getExtras();
        Double v1 = myBundle.getDouble("val1");
        Double v2 = myBundle.getDouble("val2");

        // operate on the input data
        Double vResult =  v1 + v2;

        // for illustration purposes. show data received & result
        dataReceived.setText("Data received is \n"
               + "val1= " + v1 + "\nval2= " + v2
               + "\n\nresult= " + vResult);

        // add to the bundle the computed result
        myBundle.putDouble("vresult", vResult);

        // attach updated bumble to invoking intent
        myLocalIntent.putExtras(myBundle);

        // return sending an OK signal to calling activity
        setResult(Activity.RESULT_OK, myLocalIntent);

        // experiment: remove comment
        // finish();

    }//onCreate

    @Override
    public void onClick(View v) {
           // close current screen - terminate Activity2
          finish();
    }//onClick

}//Activity2
```



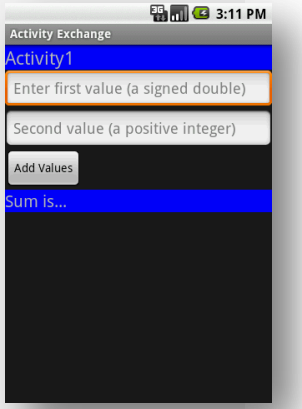
Intents

Tutorial 1. Activity Excahange *cont.*

Step5. Update the application's manifest. Add new <activity> tag for "Activity2"

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="cis493.matos.intent2b"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Activity1"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:name=".Activity2">
        </activity>

    </application>
    <uses-sdk android:minSdkVersion="4" />

</manifest>
```

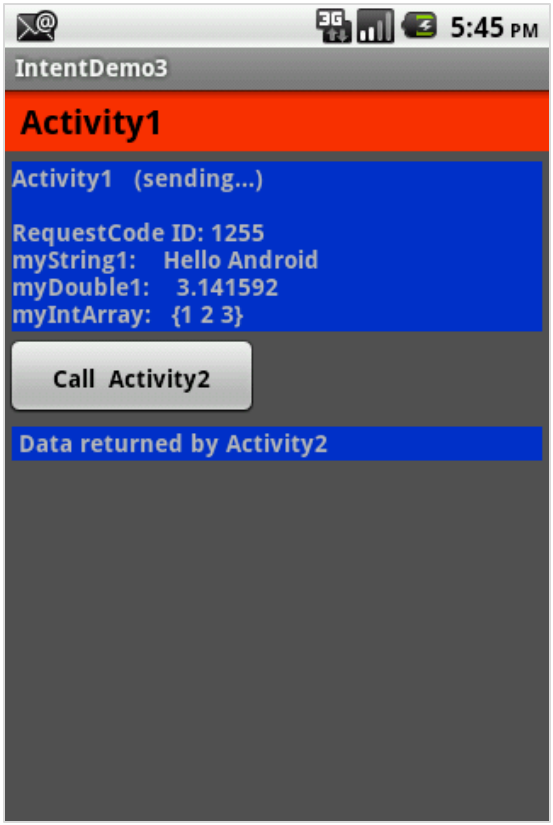
add

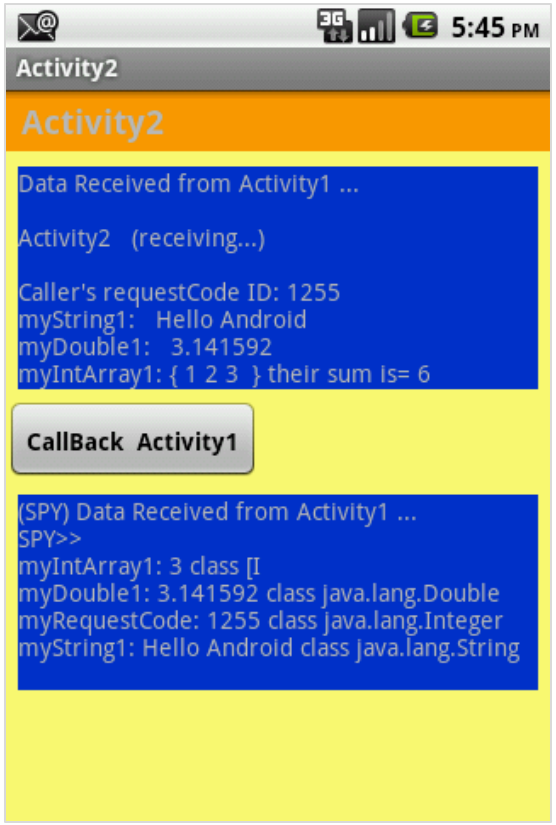


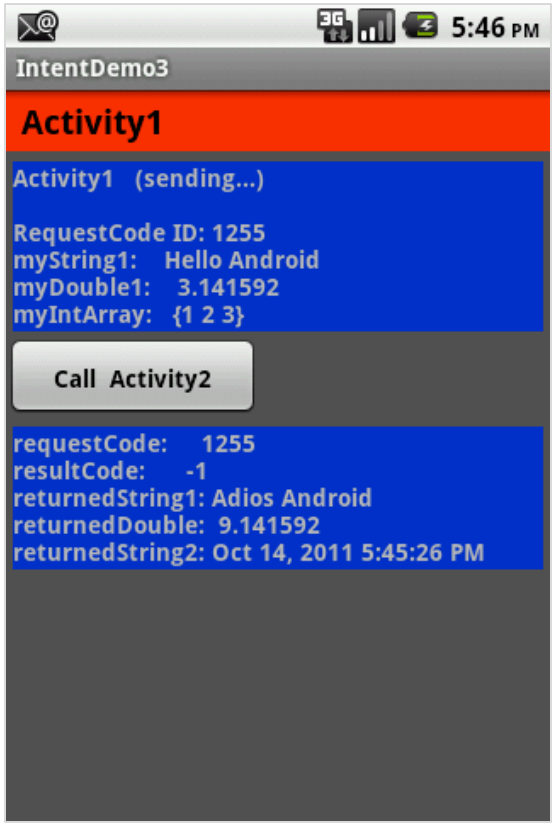
Intents

This example is similar to previous.
You may want to skip it.

Tutorial 2: : Activity1 invokes Activity2 using an Intent. A bundle conating a set of different **data types** is sent back-and-forth between both activities (see 12IntentDemo3.zip).



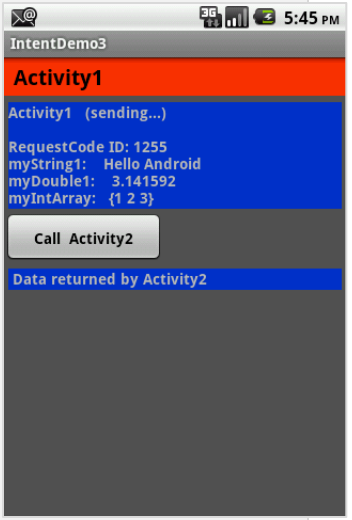




Intents

Tutorial 2 : Activity1 invokes Activity2 using an Intent. A bundle conating a set of different **data types** is sent back-and-forth between both activities (see 12IntentDemo3.zip).

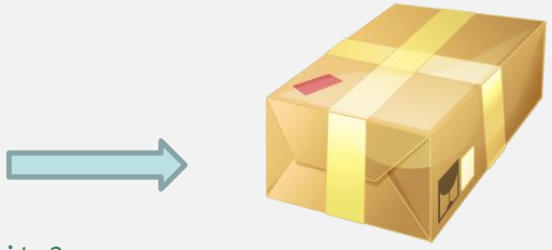
```
//Activity1: Invoking a user-defined sub-activity sending and receiving results from the sub-activity
package cis493.intents3;

import . . .;

public class Activity1 extends Activity {
   TextView label1;
   TextView label1Returned;
   Button btnCallActivity2;
   // arbitrary interprocess communication ID (just a nickname!)
   private final int IPC_ID = (int) (10001 * Math.random());

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      try {
         setContentView(R.layout.main);
         label1 = (TextView) findViewById(R.id.label1);
         label1Returned = (TextView) findViewById(R.id.label1Returned);
         btnCallActivity2 = (Button) findViewById(R.id.btnCallActivity2);
         btnCallActivity2.setOnClickListener(new Clicker1());
         // for demonstration purposes- show in top label
         label1.setText("Activity1   (sending...) \n\n"
               + "RequestCode ID: " + IPC_ID   + "\n"
               + "myString1:    Hello Android" + "\n"
               + "myDouble1:    3.141592     " + "\n"
               + "myIntArray:   {1 2 3} ");
      } catch (Exception e) {
         Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_LONG).show();
      }
   }// onCreate
```



Intents

Tutorial 2 : Activity1 invokes Activity2 using an Intent. A bundle conating a set of different **data types** is sent back-and-forth between both activities (see 12IntentDemo3.zip)

```
private class Clicker1 implements OnClickListener {
   public void onClick(View v) {
      try {
         // create an Intent to talk to Activity2
         Intent myIntentA1A2 = new Intent(Activity1.this,
                               Activity2.class);

         // prepare a Bundle and add the data pieces to be sent
         Bundle myData = new Bundle();
         myData.putInt("myRequestCode", IPC_ID);
         myData.putString("myString1", "Hello Android");
         myData.putDouble("myDouble1", 3.141592);
         int [] myLittleArray = { 1, 2, 3 };
         myData.putIntArray("myIntArray1", myLittleArray);

         // bind the Bundle and the Intent that talks to Activity2
         myIntentA1A2.putExtras(myData);

         // call Activity2 and wait for results
         startActivityForResult(myIntentA1A2, IPC_ID);

      } catch (Exception e) {
         Toast.makeText(getBaseContext(), e.getMessage(),Toast.LENGTH_LONG).show();
      }
   }// onClick
}// Clicker1
```

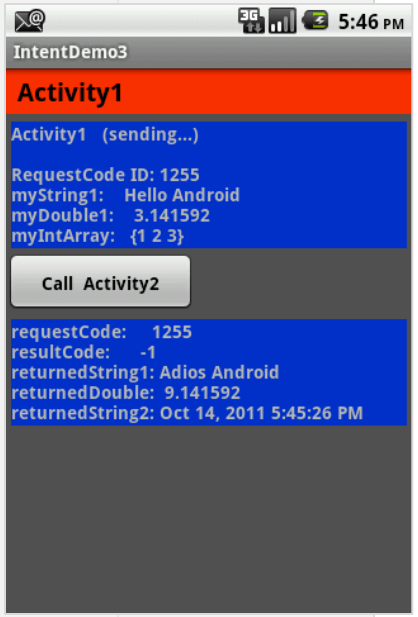


56

Intents

Tutorial 2 : Activity1 invokes Activity2 using an Intent. A bundle conating a set of different **data types** is sent back-and-forth between both activities (see 12IntentDemo3.zip).

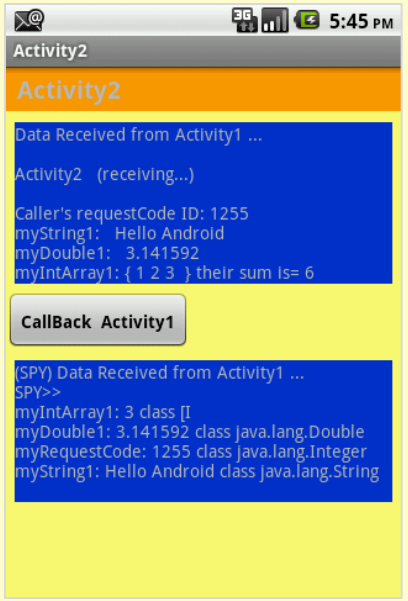
```
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
       super.onActivityResult(requestCode, resultCode, data);
       try {
          // check that these results are for us
          if (IPC_ID == requestCode) {

             // Activity2 is over - see what happened
             if (resultCode == Activity.RESULT_OK) {

                // good - we have some data sent back from Activity2
                Bundle myReturnedData = data.getExtras();
                String myReturnedString1 = myReturnedData.getString("myReturnedString1");
                Double myReturnedDouble1 = myReturnedData.getDouble("myReturnedDouble1");
                String myReturnedString2 = myReturnedData.getString("myCurrentTime");
                // display in the bottom label
                label1Returned.setText(
                     "requestCode:     " + requestCode + "\n"
                   + "resultCode:      " + resultCode  + "\n"
                   + "returnedString1: " + myReturnedString1 + "\n"
                   + "returnedDouble:  " + Double.toString(myReturnedDouble1) + "\n"
                   + "returnedString2: " + myReturnedString2);
             } else {
                // user pressed the BACK button
                label1.setText("Selection CANCELLED!");
             }// if
          }
       } catch (Exception e) {
          Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_LONG).show();
       }// try
    }// onActivityResult
 }// AndroIntent1
```



Intents

Tutorial 2: Activity2 is called to cooperate with Activity1. Data is transferred in a bundle

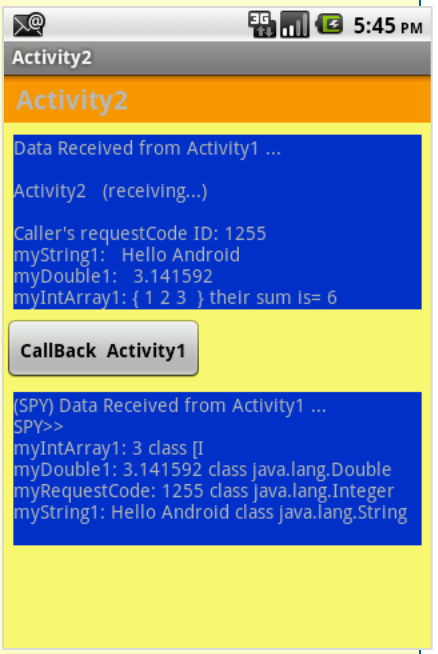
```
// Activity2. This subactivity receives a bundle of data, performs some work on the data and, returns results to Activity1.
package cis493.intents3;

import . . .;

public class Activity2 extends Activity {
    TextView label2;
    TextView spyBox;
    Button   btnCallActivity1;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main2);
        //bind UI variables to Java code
        label2 = (TextView)findViewById(R.id.label2);
        spyBox = (TextView)findViewById(R.id.spyBox);
        btnCallActivity1 = (Button)findViewById(R.id.btnCallActivity1);
        btnCallActivity1.setOnClickListener(new Clicker1());

        // create a local Intent handler – we have been called!
        Intent myLocalIntent = getIntent();

        //grab the data package with all the pieces sent to us
        Bundle myBundle = myLocalIntent.getExtras();

        //extract the individual data parts of the bundle
        int    int1 = myBundle.getInt("myRequestCode");
        String str1 = myBundle.getString("myString1");
        double dob1 = myBundle.getDouble("myDouble1");
        int[]  arr1 = myBundle.getIntArray("myIntArray1");
```



Intents

Tutorial 2: Activity2 is called to cooperate with Activity1. Data is transferred in a b

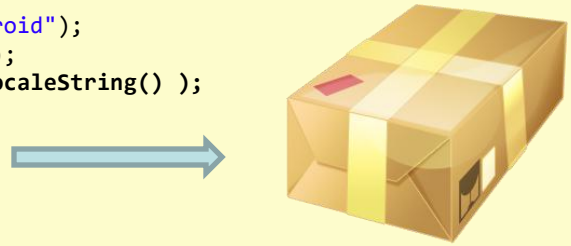
```
// +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// What if I don't know the key names?
// what came in the bundle?. This fragment shows how to use
// bundle methods to extract its data. Need to know
// ANDROID TYPES:
// class [I (array integers)
// class [J (array long)
// class [D (array doubles)
// class [F (array floats)
// class java.lang.xxx   (where xxx= Integer, Double, ...)
// +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
String spy = "\nSPY>>\n";
Set<String> myKeyNames = myBundle.keySet();
for (String keyName : myKeyNames){

          Serializable keyValue =  myBundle.getSerializable(keyName);
          String keyType = keyValue.getClass().toString();
          if (keyType.equals("class java.lang.Integer")){
                    keyValue = Integer.parseInt(keyValue.toString());
          }
          else if (keyType.equals("class java.lang.Double")){
                    keyValue = Double.parseDouble(keyValue.toString());
          }
          else if (keyType.equals("class java.lang.Float")){
                    keyValue = Float.parseFloat(keyValue.toString());
          }
          else if (keyType.equals("class [I")){
                    int[] arrint = myBundle.getIntArray(keyName);
                    int n = arrint.length;
                    keyValue = arrint[n-1]; // show only the last!
          }
          else {
          keyValue = (String)keyValue.toString();
          }
          spy += keyName + ": " + keyValue +  " " + keyType + "\n" ;
}
spyBox.append(spy);
```



Intents

Tutorial 2: Activity2 is called to cooperate with Activity1. Data is transferred in a b

```
        // ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         //do something with the data here (for example...)
         String strArr  = "{ ";
         int sumIntValues = 0;
         for (int i=0; i<arr1.length; i++) {
          sumIntValues += arr1[i];
          strArr += Integer.toString( arr1[i] ) + " ";
         }
         strArr += " } their sum is= " + sumIntValues;

         //show arriving data in GUI label2
         label2.append("\n\nActivity2   (receiving...) \n\n" +
                   "Caller's requestCode ID: " + int1 + "\n" +
                    "myString1:    " + str1 + "\n" +
                    "myDouble1:    " + Double.toString(dob1) + "\n" +
                    "myIntArray1: " + strArr);

         //now go back to myActivity1 with some new data made here
         double someNumber = sumIntValues + dob1;
         myBundle.putString("myReturnedString1", "Adios Android");
         myBundle.putDouble("myReturnedDouble1", someNumber);
         myBundle.putString("myCurrentTime", new Date().toLocaleString() );
         myLocalIntent.putExtras(myBundle);

         // all done!
         setResult(Activity.RESULT_OK, myLocalIntent);

     }//onCreate

     private class Clicker1 implements OnClickListener {
       public void onClick(View v) {
          //clear Activity2 screen so Activity1 could be seen
          finish();
       }//onClick
     }//Clicker1

}//Activity2
```



Intents

Tutorial 2: Activity2 is called to cooperate with Activity1. Data is transferred in a bundle

Layout: main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:id="@+id/linLayout"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android"
android:background="#ff555555">
<TextView
android:id="@+id/caption1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ffff3300"
android:padding="4sp"
android:text=" Activity1 "
android:textSize="20px"
android:textStyle="bold"
android:textColor="#ff000000"
>
</TextView>
<TextView
android:id="@+id/widget107"
android:layout_width="fill_parent"
android:layout_height="2sp"
>
</TextView>
<TextView
android:id="@+id/label1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff0033cc"
android:text="Data to be sent to SubActivity:"
android:layout_margin="4dip"
android:textStyle="bold|normal">
</TextView>
<Button
android:id="@+id/btnCallActivity2"
android:layout_width="149px"
android:layout_height="wrap_content"
android:text="Call  Activity2"
android:textStyle="bold"
android:padding="6sp"
>
</Button>
<TextView
android:id="@+id/label1Returned"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff0033cc"
android:text=" Data returned by Activity2"
android:layout_margin="4dip"
android:textStyle="bold|normal">
</TextView>
</LinearLayout>
```

Intents

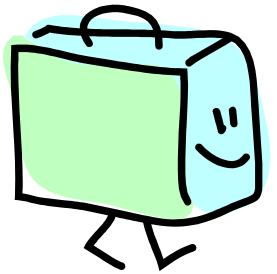
Tutorial 2: Activity2 is called to cooperate with Activity1. Data is transferred in a bundle

Layout: main2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:id="@+id/linearLayout"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android"
android:background="#ffffff77">
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ffff9900"
android:padding="4sp"
android:text=" Activity2"
android:textSize="20px"
android:textStyle="bold"
>
</TextView>
<TextView
android:id="@+id/widget107"
android:layout_width="fill_parent"
android:layout_height="2sp"
>
</TextView>
<TextView
android:id="@+id/label2"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff0033cc"
android:text="Data Received from Activity1 ..."
android:textStyle="normal"
android:layout_margin="7dip">
</TextView>
<Button
android:id="@+id/btnCallActivity1"
android:layout_width="149px"
android:layout_height="wrap_content"
android:padding="6sp"
android:text="CallBack  Activity1"
android:textStyle="bold"
>
</Button>

<TextView
android:id="@+id/spyBox"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff0033cc"
android:text="(SPY) Data Received from Activity1 ..."
android:textStyle="normal"
android:layout_margin="7dip">
</TextView>
</LinearLayout>
```


62

Intents



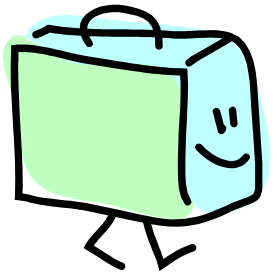
Appendix A. Bundling Complex Objects

Extending Tutorial2 to allow Activity1 to create a local object and pass it to Activity2 into the IPC bundle as serialized data.

Step 1. Create an Object. Make sure it implements **Serializable** interface.

```
package cis493.intents3;

import java.io.Serializable;

public class Person implements Serializable  {
          private static final long serialVersionUID = 1L;

          private String firstName;
          private String lastName;

          public Person(String firstName, String lastName) {
                    super();
                    this.firstName = firstName;
                    this.lastName = lastName;
          }

          public String getFullName() {
                    return  firstName + " " + lastName;
          }
}//Person
```

Intents



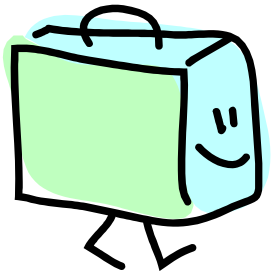
Appendix A. Bundling Complex Objects

Extending Tutorial2 to allow Activity1 to create a local object and pass it to Activity2 into the IPC bundle as serialized data.

Step 2. Modify Activity1. Create an instance of the Person class and add it to the bundle using the method *putSerializable* (*key, object*);

```
// prepare a Bundle and add the data pieces to be sent
Bundle myData = new Bundle();
. . .

// creating an object and passing it into the bundle
Person p1 = new Person("Maria", "Macarena");
myData.putSerializable("person", p1);

// bind the Bundle and the Intent that talks to Activity2
myIntentA1A2.putExtras(myData);

// call Activity2 and wait for results
startActivityForResult(myIntentA1A2, IPC_ID);
```

Intents

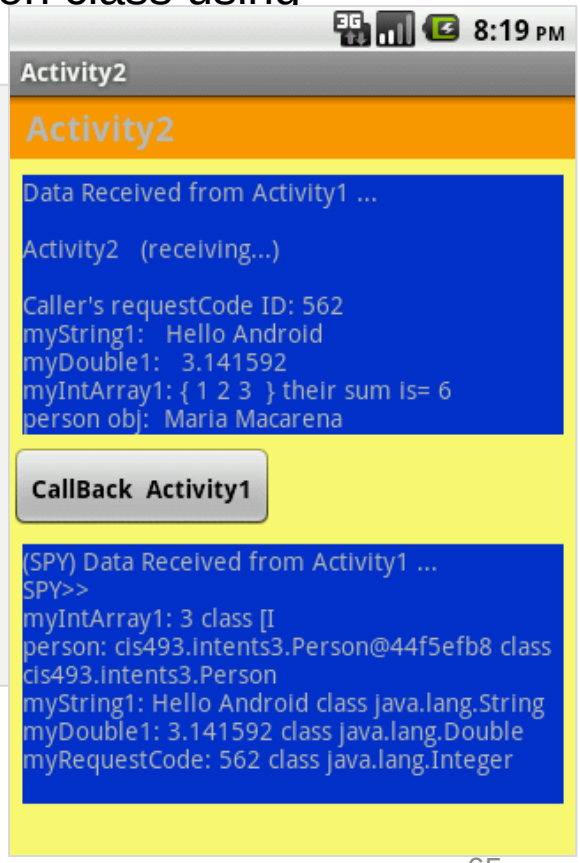


Appendix A. Bundling Complex Objects

Extending Tutorial2 to allow Activity1 to create a local object and pass it to Activity2 into the IPC bundle as serialized data.

Step 3. Modify Activity2. Capture the instance of the Person class using the method *getSerializable* (*key*);

```
 // create a local Intent handler – we have been called!
Intent myLocalIntent = getIntent();

 //grab the data package with all the pieces sent to us
 Bundle myBundle = myLocalIntent.getExtras();

 //extract the individual data parts of the bundle
 . . .

 Person p = (Person) myBundle.getSerializable("person");
 String pval = p.getFullName();

 . . .
```



Note: The object person has a complex class type received as:

```
class packageName.Person
```

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. The caller however continues to execute in its own thread.

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.

67

Intents

Starting Activities and Getting Results

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

startActivityForResult (Intent, requestCodeID **)**

Where *requestCodeID* is an arbitrary value you choose to identify the call (similar to a 'nickname').

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

onActivityResult (requestCodeID, resultCode, Intent **)**

68

Intents

Starting Activities and Getting Results

- Before an invoked activity exits, it can call **setResult (resultCode)** to return a termination signal back to its parent.
- It is convenient to 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

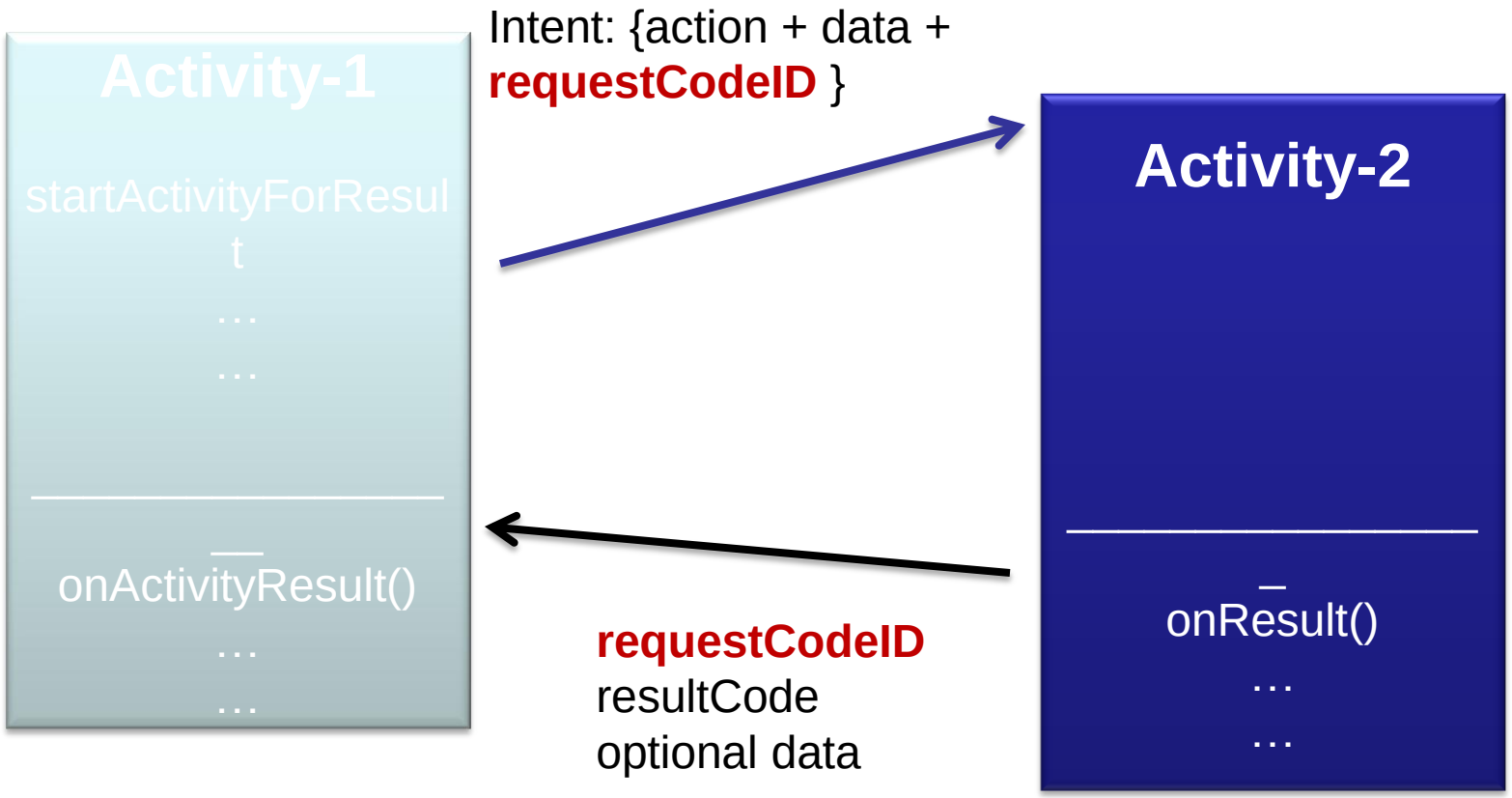
```
onActivityResult (int requestCodeID, int resultCode, Intent data)
```

- If a child activity fails for any reason (such as crashing), the parent activity will receive a result with the code **RESULT_CANCELED**.

69

Intents

Starting Activities and Getting Results

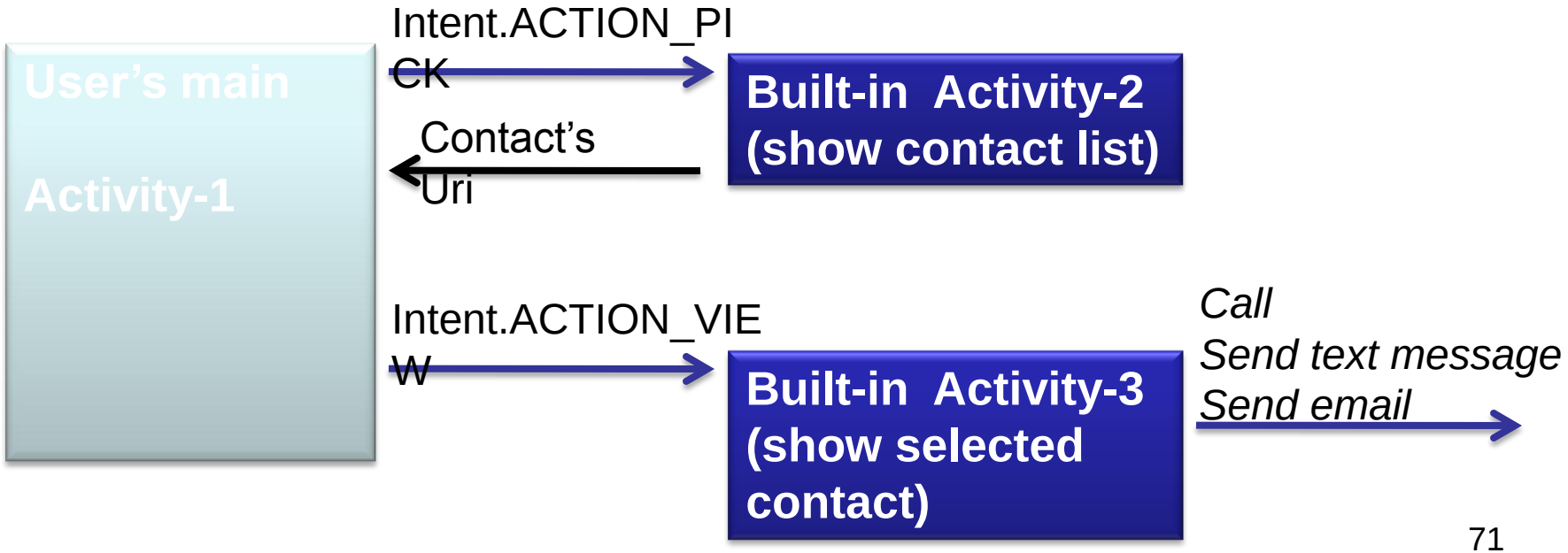


70

Intents

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

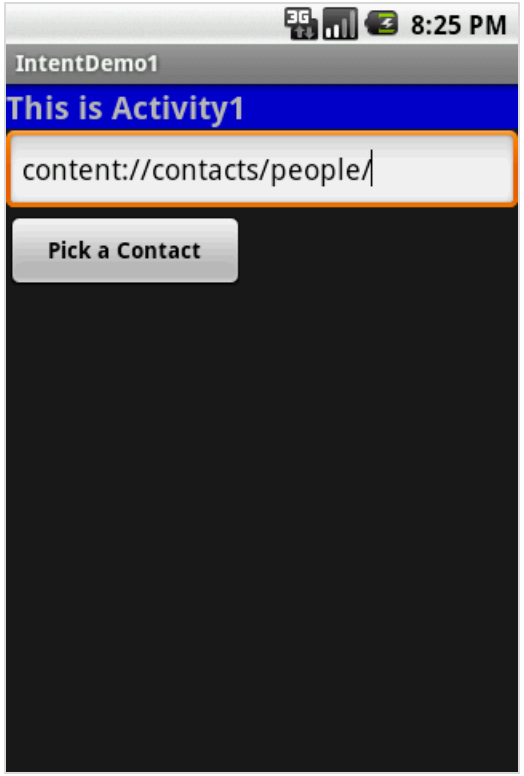
1. Show all contacts and pick a particular one (*Intent.ACTION_PICK*).
2. For a successful interaction the main-activity accepts the returned URI identifying the person we want to call (*content://contacts/people/n*).
3. 'Nicely' show the selected contact's entry allowing calling, texting, emailing actions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

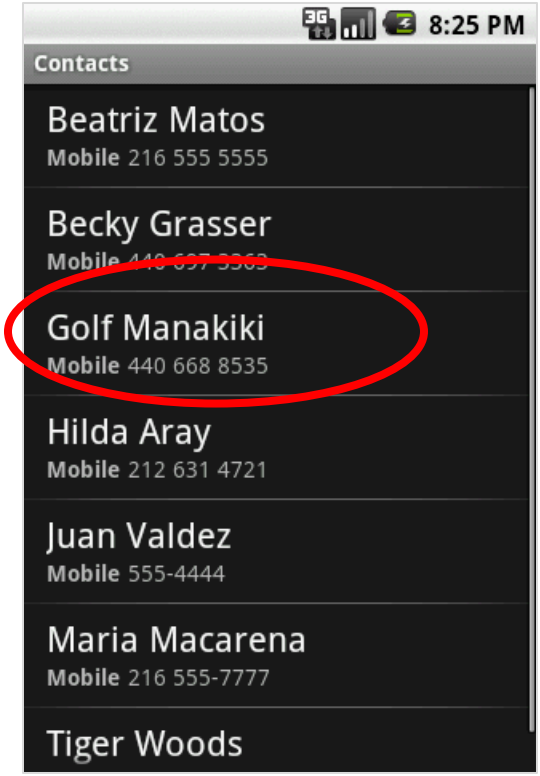


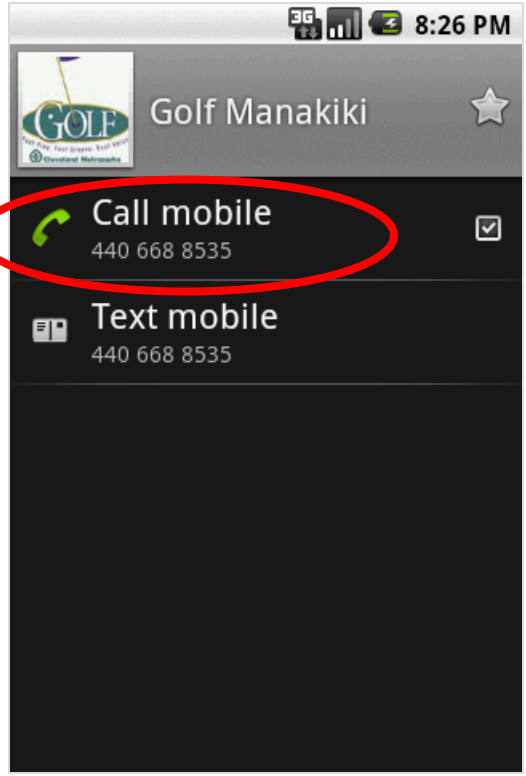
71

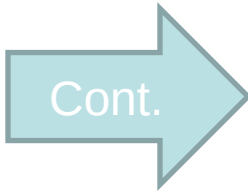
Intents

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









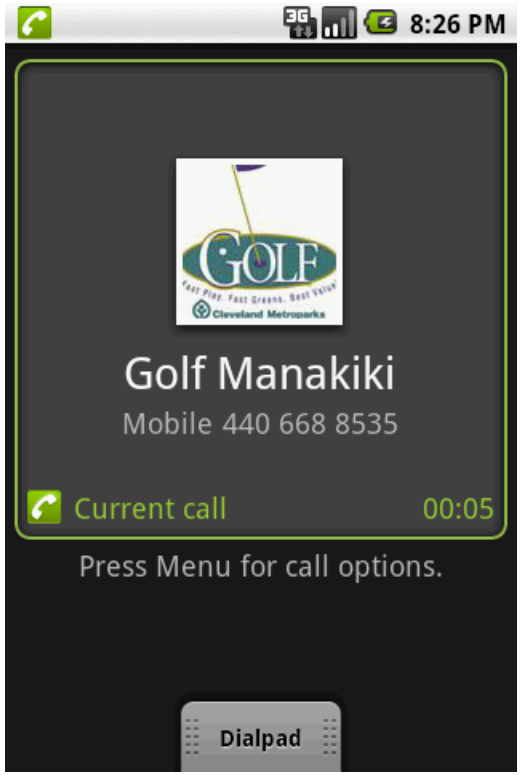
Main Activity

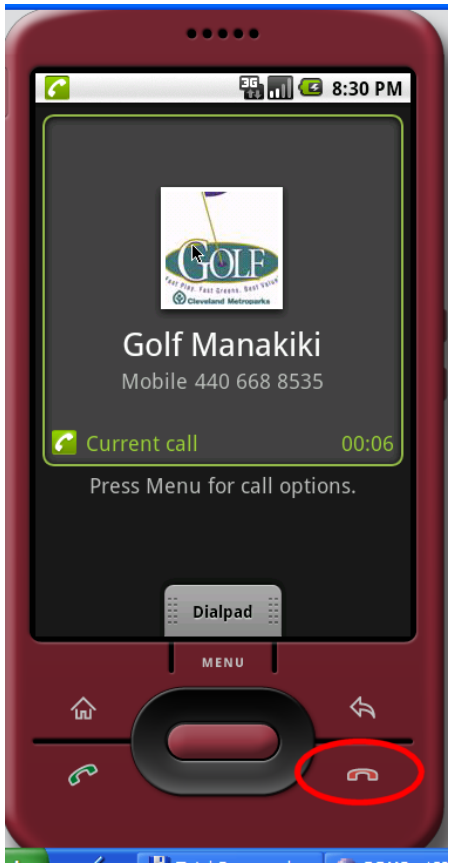
Intent.ACTION_PICK

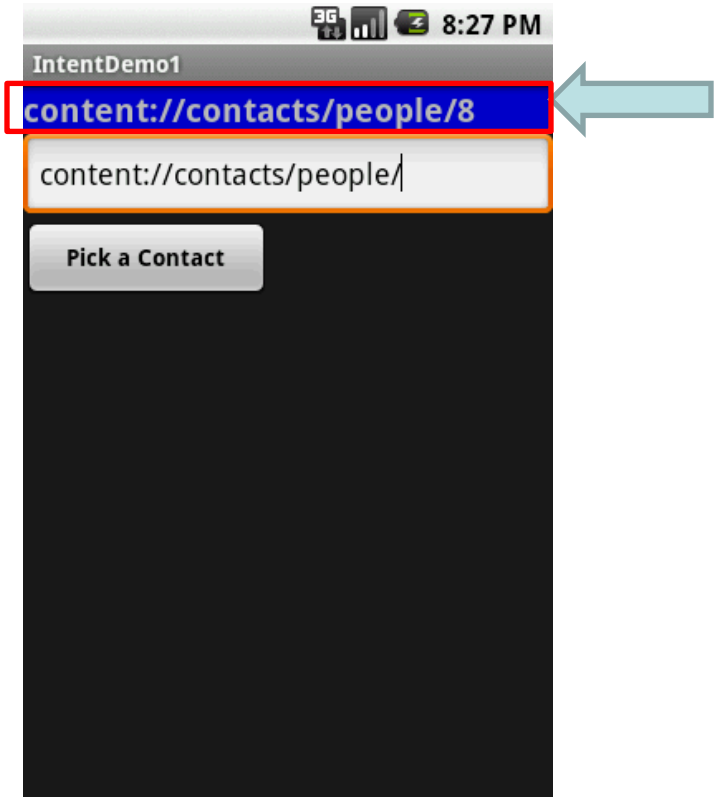
Intent.ACTION_VIEW

Intents

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







Place the call

Terminate the call

Selected contact's URI

73

Intents

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

```
//IntentDemo2_Intent: making a phone call
//receiving results from a sub-activity
package cis493.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 IntentDemo2 extends Activity {
    TextView label1;
    EditText text1;
    Button   btnCallActivity2;
```

Intents

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

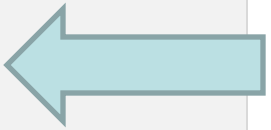
```
@Override
   public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   try {
            setContentView(R.layout.main);
            label1 = (TextView)findViewById(R.id.label1);
            text1 = (EditText)findViewById(R.id.text1);

            btnCallActivity2 = (Button)findViewById(R.id.btnPickContact);
            btnCallActivity2.setOnClickListener(new ClickHandler());
   }
   catch (Exception e) {
             Toast.makeText(getBaseContext(),
             e.getMessage(), Toast.LENGTH_LONG).show();
   }
 }//onCreate
```


75

Intents

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

```
private class ClickHandler implements OnClickListener {
    @Override
    public void onClick(View v) {
        try {
             // myData refer 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
```



Intents

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


```
@Override
protected void onActivityResult(int requestCode,
                                int resultCode,
                                Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    try {
        // use requestCode to 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);
                }
```

Intents

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

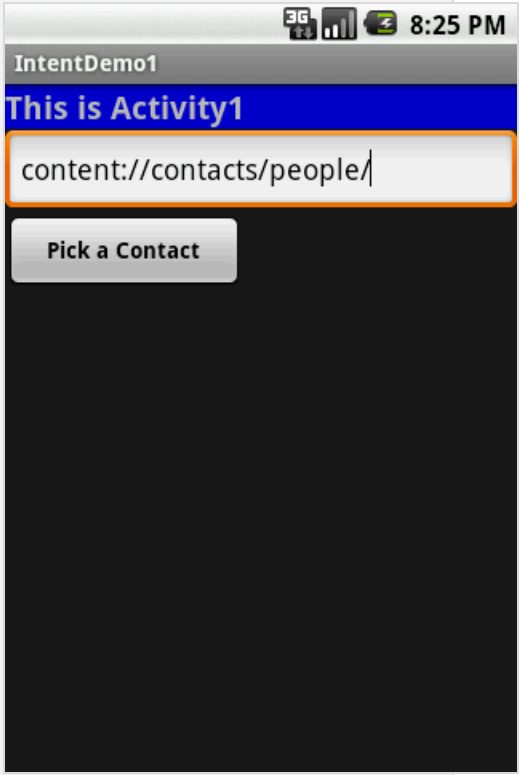
```
                    else {
                         //user pressed the BACK button
                         label1.setText("Selection CANCELLED "
                                        + requestCode + " " + resultCode);
                         }
                         break;
                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

}//IntentDemo2
```


78

Intents

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

```
<?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="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>
```



Intents

Example3. Showing Pictures and Video - Calling a sub-activity, receiving results

```
private void showSoundTracks() {

  Intent myIntent = new Intent();
  myIntent.setType("video/*, images/*");
  myIntent.setAction(Intent.ACTION_GET_CONTENT);
  startActivityForResult(myIntent, 0);

}//showSoundTracks


@Override
protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
  super.onActivityResult(requestCode, resultCode, intent);

  if ((requestCode == 0) && (resultCode == Activity.RESULT_OK)) {

      String selectedImage = intent.getDataString();

      Toast.makeText(this, selectedImage, 1).show();

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

All videos and all still images

Intents

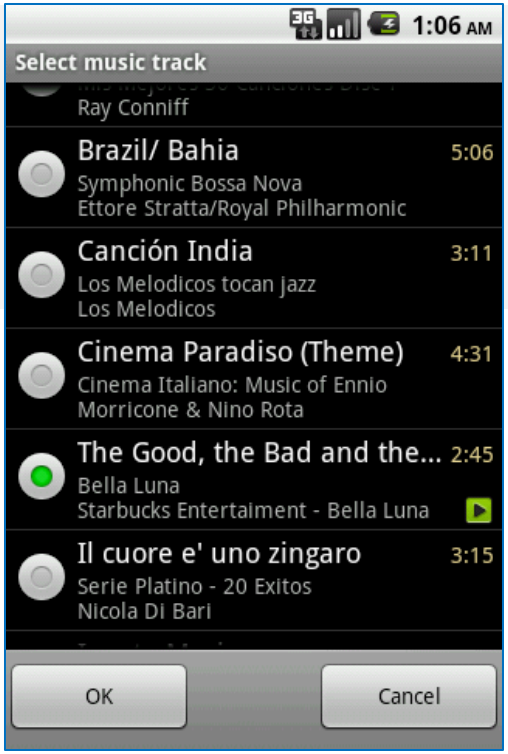
Example3. Showing Pictures and Video - Calling a sub-activity, receiving results.

Intents

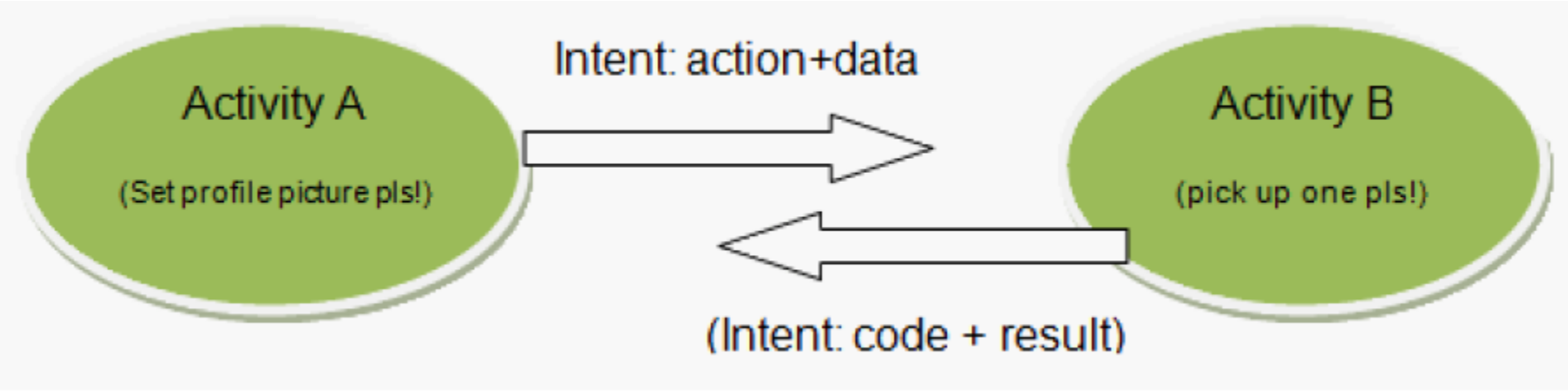
Example4. Showing/Playing Sound Tracks - Calling a sub-activity, receiving

```
private void showSoundTracks() {
   Intent myIntent = new Intent();
   myIntent.setType("audio/mp3");
   myIntent.setAction(Intent.ACTION_GET_CONTENT);
   startActivityForResult(myIntent, 0);
}//showSoundTracks
```

The returned string value is similar to the following
“content://media/external/audio/media/14”
ACTION_VIEW on that Uri would produce a result
similar to the image on the right



82


Activity A
(Set profile picture pls!)
Intent: action+data
Activity B
(pick up one pls!)
(Intent: code + result)

Intent Filters

INTENTS

- Intent là một mô tả trừu tượng của 1 hành động được thực hiện.
- Là một thành phần quan trọng trong quá trình khởi tạo 1 activity.
- Thông tin chủ yếu trong intent : *action* & *data*.

ACTION	DATA	Misc
The general action to be performed, such as: ACTION_EDIT, ACTION_VIEW, ACTION_MAIN, ACTION_LAUNCHER etc.	The data to operate on, such as a person record in the contacts database, expressed as a **URI**. *I am good for editing a document* *I am good for viewing a document* *I am the first exec. Activ. of Application* *Put me on the phone's Menu_Pad*	

Source: http://developer.android.com/reference/android/content/Intent.html

Intent Filters

Parts of a Typical Intent

ACTION

Standard

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

ACTION_TIME_TICK
ACTION_TIME_CHANGED
ACTION_TIMEZONE_CHANGED
ACTION_BOOT_COMPLETED
ACTION_PACKAGE_ADDED
ACTION_PACKAGE_CHANGED
ACTION_PACKAGE_REMOVED
ACTION_PACKAGE_RESTARTED
ACTION_PACKAGE_DATA_CLEARED
ACTION_UID_REMOVED
ACTION_BATTERY_CHANGED
ACTION_POWER_CONNECTED
ACTION_POWER_DISCONNECTED
ACTION_SHUTDOWN

DATA

URI

CONTENTS *such as:*

content://contacts/
content://contacts/1

SCHEME *such as:*

tel:123
http://aaa.bbb.ccc
mailto://aa@bbb.ccc
ftp://aaa.bbb.ccc
. . .
pop://
smtp://
ssl://

MISC

Category

CATEGORY_DEFAULT
CATEGORY_BROWSABLE
CATEGORY_TAB
CATEGORY_ALTERNATIVE
CATEGORY_SELECTED_ALTERNATIVE
CATEGORY_LAUNCHER
CATEGORY_INFO
CATEGORY_HOME
CATEGORY_PREFERENCE
CATEGORY_TEST

MIME

Explicit type (a MIME type) of the intent data.

Component

Explicit name of a component class to use for the intent.

Extras

putExtra(String, Bundle)

Flags

86

Intent Filters

Aside: MIME

" … This set of documents, collectively called the Multipurpose Internet Mail Extensions, or MIME, redefines the format of messages to allow for

(1) textual message bodies in character sets other than US-ASCII,

(1) an extensible set of different formats for non-textual message bodies,

(2) multi-part message bodies, and

(3) textual header information in character sets other than US-ASCII."

NOTE:
Current usage of MIME describes content type in general.

Source: Multipurpose Internet Mail Extensions. (MIME) Part Two: Media Types. Available at: http://tools.ietf.org/html/rfc2046

87

Intent Filters

Intent Resolution

Khi intent gửi ra 1 yêu cầu, Android sẽ tím kiếm các trả lời phù hợp cho intent đó.

Để quyết định intent nào thực hiện dưa trên mô tả của intent, và đây là cách chia intent thành 2 thành phần :

Intent tường minh(Explicit Intents) dùng để nói đến một thành phần cụ thể (setComponent(ComponentName) or setClass(Context, Class)), 1 class cụ thể để thực hiện. Đây là 1 cách để gọi 1 activity khác thực thi.

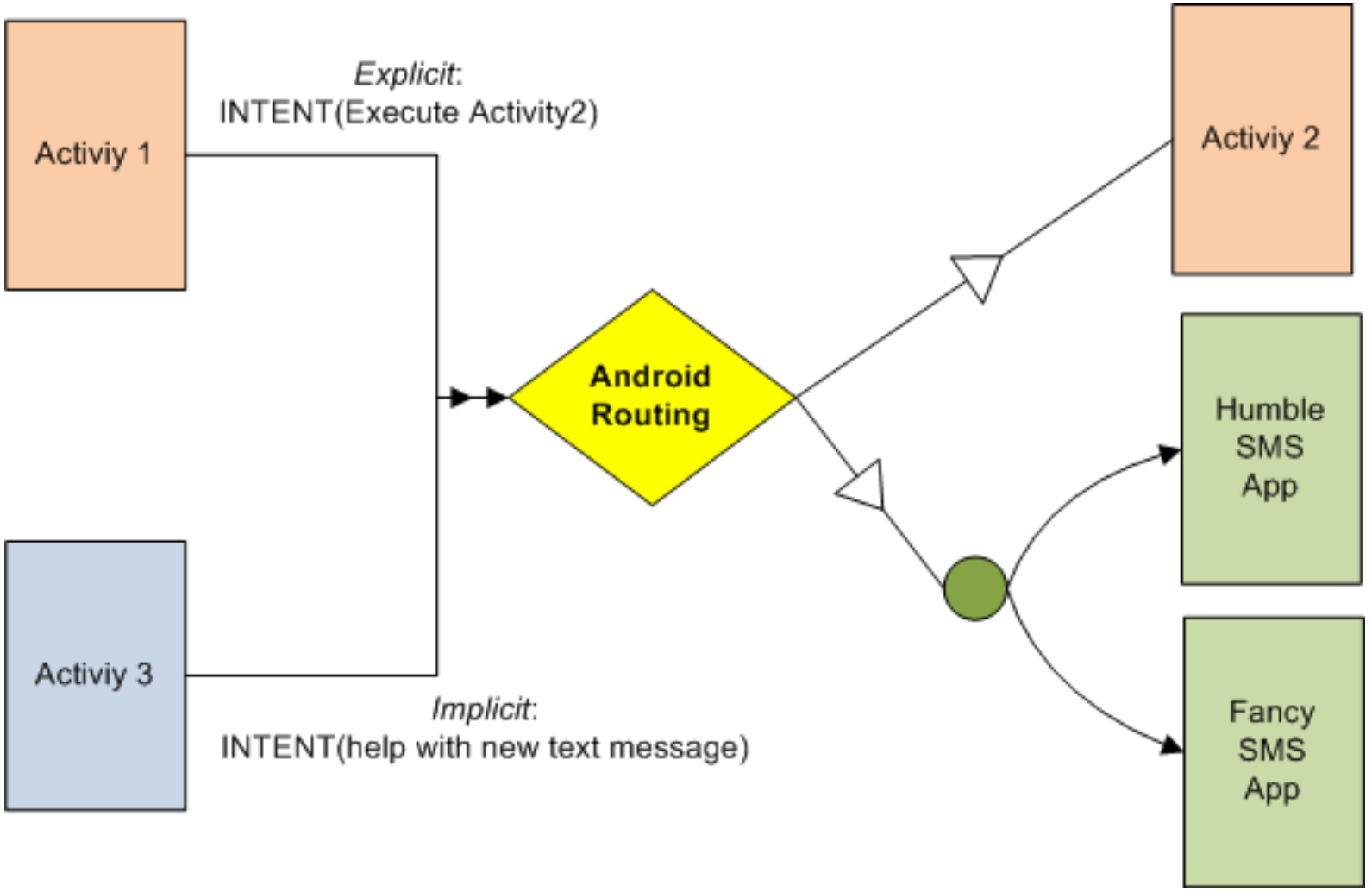
Intent không tường minh (Implicit Intents) không phải là một thành phần cụ thể (1 class). Nhưng các intent này đủ thông tin, giúp hệ thống xác định thành phần nào được thực thi.

<intent-filter>

88

Intent Filters

Intent Resolution



Intent Filters

Intent Resolution

Activity3 gửi 1 yêu cầu xử lý tin nhắn vừa được gửi tới.

Giả sử người dùng đã cài đặt ứng dụng “Fancy SMS” để thay thế 1 ứng dụng có sẵn trong hệ thống “HUMBLE SMS”.

Upon the arrival of the implicit Intent, Android will (somehow) tell the user:

You have got a new text-message. I have a FANCY and a HUMBLE SMS application – which one you want me to execute? Make it a default?

Choosing candidates: For an activity to be eligible for execution it must:

1. Support the specified action
2. Support the indicated MIME type (if supplied)
3. Support all of the *categories* named in the intent.

RULE OF THUMB: *Your Intents should be as specific as possible*

90

Intent Filters

Example: Intent Filters

The Manifest tells the application (**FancySms**) is able to intercept incoming SMS data using its ***SMSReceiver*** (potential alternative to the default SMS app.)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cis493.intentfilters"  android:versionCode="1"  android:versionName="1.0.0">
   <uses-permission android:name="android.permission.RECEIVE_SMS"  />
   <application android:icon="@drawable/icon"  >
      <activity  android:name=".FancySms"  >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
      <receiver android:name="SMSReceiver"  android:enabled="true"  >
            <intent-filter>
                  <action android:name="android.provider.Telephony.SMS_RECEIVED"  />
            </intent-filter>
      </receiver>
   </application>
</manifest>
```

Intent Filters

Comments on the example:

- The application consists of two components:

1. a common Activity called **FancySms** (acting as the main routine) and
2. a background Service (BroadcastReceiver) called **SMSService**.

- The clause below indicates the application is allowed to receive SMS

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

- The component **SMSService** has the filter

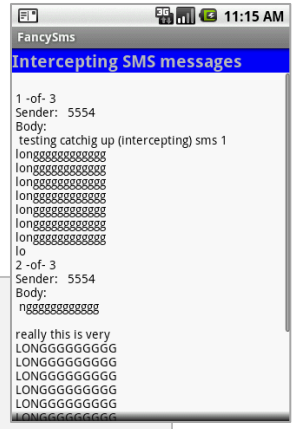
```
<intent-filter>
        <action android:name="android.provider.Telephony.SMS_RECEIVED"    />
</intent-filter>
```

that triggers its execution whenever a new SMS is received

- Other applications with the same filter can be also called by Android when new SMS arrives (until a DEFAULT is chosen)

92

Intent Filters



Example: Intercepting Incoming SMS

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
     android:id="@+id/mainLayout"
     android:layout_width="fill_parent" android:layout_height="fill_parent"
     android:orientation="vertical"
     xmlns:android="http://schemas.android.com/apk/res/android"
>
  <TextView
     android:layout_width="fill_parent"   android:layout_height="wrap_content"
     android:textSize="20px"  android:textStyle="bold" ndroid:background="#ff0000ff"
     android:text="Intercepting SMS messages"
     />
  <ScrollView
     android:id="@+id/myScroller1"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     >
     <TextView
          android:id="@+id/theMessage"
          android:layout_width="fill_parent"  android:layout_height="fill_parent"
          android:background="#ffffffff"      android:padding="4px"
          android:textSize="14px"             android:textColor="#ff000000"
          />
  </ScrollView>
</LinearLayout>
```

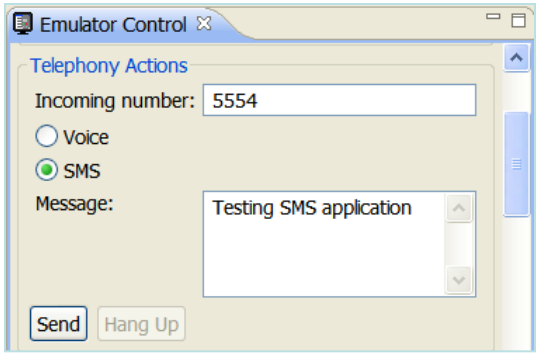
Intent Filters

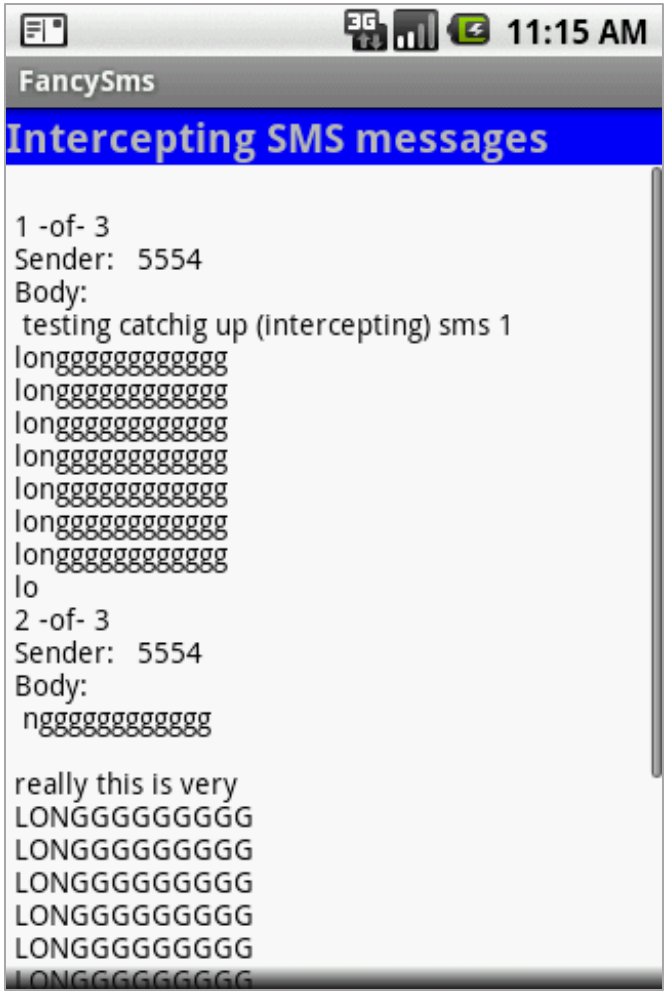
Example: Intercepting Incoming SMS

DDMS

Note:
Test the following application from the Eclipse's **DDMS** perspective. Select "Emulator Control" > "Telephony Actions". Set phone no. to 5554, type a message, click on Send.

Alternatively you may start another emulator and send SMS to 5554





94

Intent Filters

Example: Intercepting Incoming SMS

```
// FancySms: main screen - displays intercepted SMS
package cis493.intentfilters;

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

public class FancySms extends Activity {
    static TextView txtMsg;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        txtMsg = (TextView)findViewById(R.id.theMessage);
    }
}// class FancySms
```


95

Intent Filters

Example: Intercepting Incoming SMS

```
// SMSReceiver:  listens to broadcasted SMS_RECEIVED signals
package cis493.intentfilters;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.gsm.SmsMessage;
import android.widget.Toast;

public class SMSReceiver extends BroadcastReceiver {

@Override
public void onReceive(Context context, Intent intent) {
// Android saves in a bundle the current text-message
// under name "pdus" and type: Object[]. Later we cast to
// SmsMessage[]. Jargon pdu stands for "protocol data unit"
Bundle bundle = intent.getExtras();
```


96

Intent Filters

Example: Intercepting Incoming SMS

```
    Object messages[] = (Object[]) bundle.get("pdus");
    SmsMessage smsMessage[] = new SmsMessage[messages.length];

    // Note: long sms are broken and transmitted into various pieces
    String msg = "";
    int smsPieces = messages.length;

    for (int n = 0; n < smsPieces; n++) {
     smsMessage[n] = SmsMessage.createFromPdu((byte[]) messages[n]);
     // grab all pieces of the intercepted sms
     msg  += "\n" + (n + 1) + " -of- " + smsPieces + "\n"
          + "Sender:\t"   + smsMessage[n].getOriginatingAddress() + "\n"
          + "Body:  \n " + smsMessage[n].getMessageBody();
    }

    // show first part of intercepted (current) message
    Toast toast = Toast.makeText(context, "FANCY >>> Received SMS: "
                + smsMessage[0].getMessageBody(), Toast.LENGTH_LONG);
    toast.show();

    cis493.intentfilters.FancySms.txtMsg.setText(msg);
    }
}// class SMSReceiver
```