Handheld Application Development

Lec 4: Intent

Ekarat Rattagan, PhD

Intents

An Android application could include any number of activities

- An activity uses the setContentView( ) to expose 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.

Intents

- An asynchronous message.
- Bind individual components to each other at runtime.
- Think of them as the messengers that request an action from other components, whether the component belongs to your app or another.

Intents

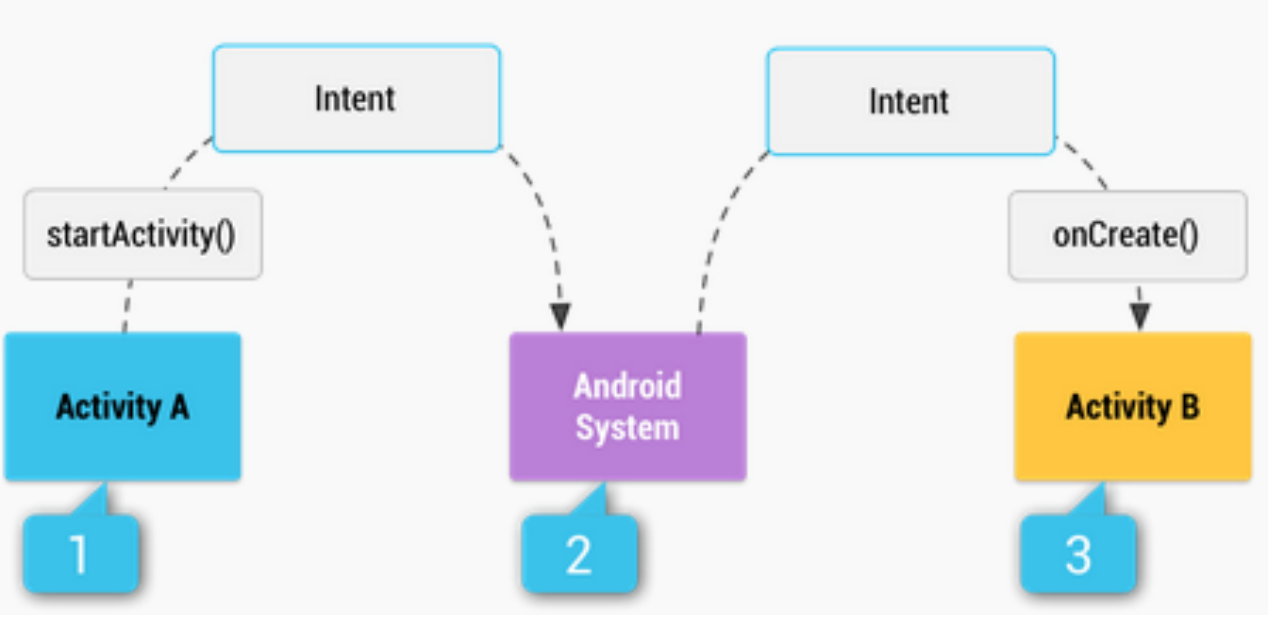


Figure 1. Illustration of how an implicit intent is delivered through the system to start another activity:

1. *Activity A* creates an Intent with an action description and passes it to startActivity().
2. The Android System searches all apps for an intent filter that matches the intent.
3. When a match is found, The system starts the matching activity (*Activity B*) by invoking its onCreate() method and passing it the **Intent**.

Intents

Public constructors
`Intent()` Create an empty intent.
`Intent(Intent o)` Copy constructor.
`Intent(String action)` Create an intent with a given action.
`Intent(String action, Uri uri)` Create an intent with a given action and for a given data url.
`Intent(Context packageContext, Class<?> cls)` Create an intent for a specific component.
`Intent(String action, Uri uri, Context packageContext, Class<?> cls)` Create an intent for a specific component with a specified action and data.

Action/Data

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

startActivity (myActivity);
```

Built-in or user-created activity

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

Type of Intents

1. Explicit intents

Specify the Android component to be called, e.g., Activity B and A, e.g.,

```
Intent intent1 = new Intent(this, ActivityB.class);
startActivity(intent1);
```

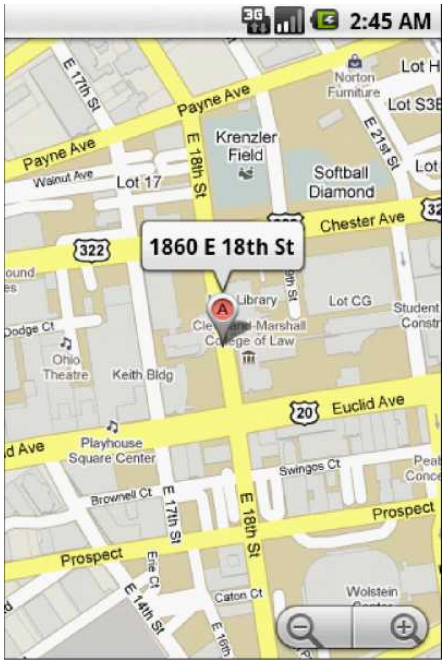
2. Implicit intents

Do not directly specify the Android components which should be called, e.g.,

```
Intent intent2 = new Intent(Intent.ACTION_VIEW,
                Uri.parse("http://www.google.com"));
startActivity(intent2);
```

//Uri: A Uniform Resource Identifier (URI) is a compact sequence of characters that identifies an abstract or physical resource.

Intents



Example 1.1 : Using Standard Actions

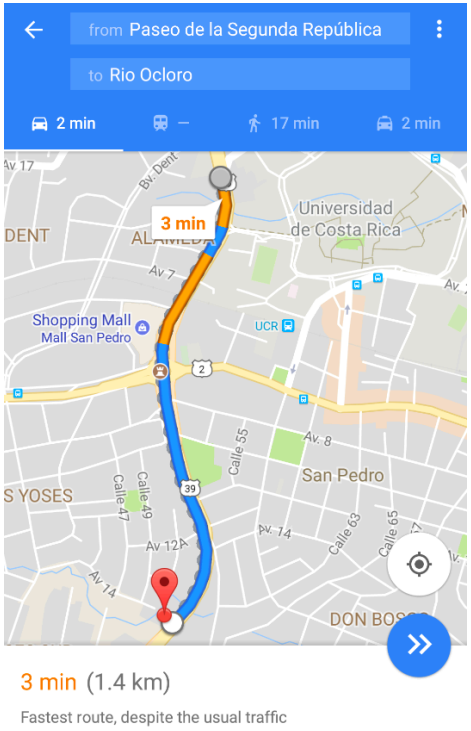
Geo Mapping an Address

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

```
String geoCode = "geo:0,0?q=18 60+east+18th+street+cleveland+oh";
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(geoCode));
startActivity(intent);
```

```
* geCode can also be Latitude/Longtitude
String geoCode = ""geo:41.5020952,-81.6789717";
```

Intents

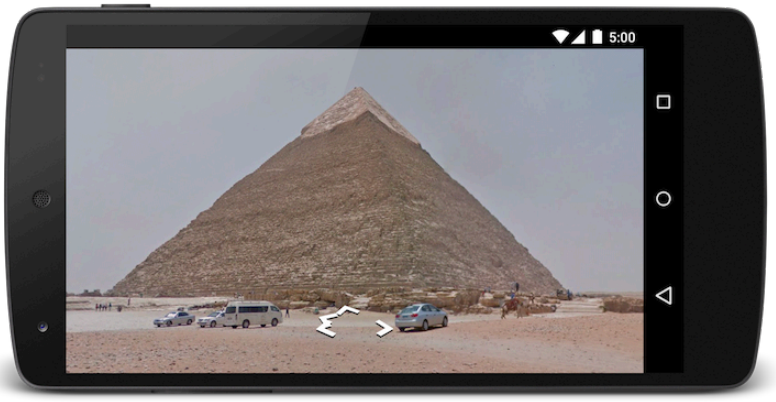


GeCode can be different format

Example 1.2: Getting directions from position A to B

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

Intents



Geo Mapping - Google StreetView

cbp: to control a camera's orientation.
The most significant values are the second, fourth and fifth which set the bearing, zoom and tilt respectively. The first and third values are not supported, and should be set to 0.
cbp=a,b,c,d,e

a = 0
b = (bearing) 0 - 360 degree (0 is North, 90 is East)
c = 0
d = (zoom), 0 is default, 1 is double magnification
e = (tilt) -90 (Up), 0 centered on the horizontal, 90(Down)

```
Example 1.3:String geoCode =
"google.streetview:cbll=29.9774614,31.1329645&cbp=0,30,0,0,-15";
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(geoCode));
startActivity(intent);
```

Warning: Process: com.example.ekaratrattagan.mobileapp, android.content.ActivityNotFoundException: No Activity found to handle Intent

More details of google streetview: https://developers.google.com/maps/documentation/urls/android-intents

More Action/Data

Examples of **action / data** pairs are:
ACTION_DIAL `tel:555-1234`
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** ***(deprecated)***
Edit information about the person whose identifier is ”2”.

ACTION_VIEW **content://contacts/people/2** ***(deprecated)***
Used to start an activity to display *2-nd* person.

ACTION_VIEW ***content://contacts/*** **people/ (deprecated)**
Display a list of people, which the user can browse through. Selecting a particular person to view would result in a new intent

More Action/Data

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_SEND **ACTION_SENDTO**	ACTION_FACTORY_TEST

Send values between Activities

Send values between Activities

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:

putExtra: Send data between activities

Activity 1 (Sender)

```
int data = 5;
Intent intent = new Intent(getApplicationContext(),
ActivityB.class);
intent.putExtra("my_var_name", data);
startActivity(intent);
```

Activity 2 (Receiver)

```
Intent intent = getIntent();
int temp = intent.getIntExtra("my_var_name", 0);
```

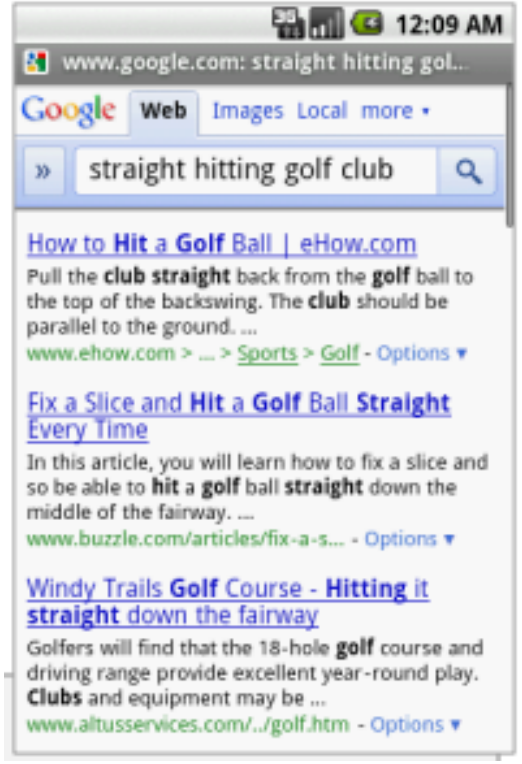
Send values between Activities

Data type of extra

- boolean
  - boolean[]
  - byte
  - byte[]
  - char
  - char[]
  - CharSequence
  - CharSequence[]
  - double
  - double[]
  - float
  - float[]
  - int
- int [ ]
  - long
  - long[]
  - short
  - short[]
  - String
  - String[]
  - ArrayList<CharSequence>
  - ArrayList<String>
  - ArrayList<Integer>
  - Parcelable
  - Serializable

Intents - Secondary Attributes

Not only defined variables, but also built-in variables



Example 1.4 : 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

Example 1.5 : Sending a text message (using extra attributes)

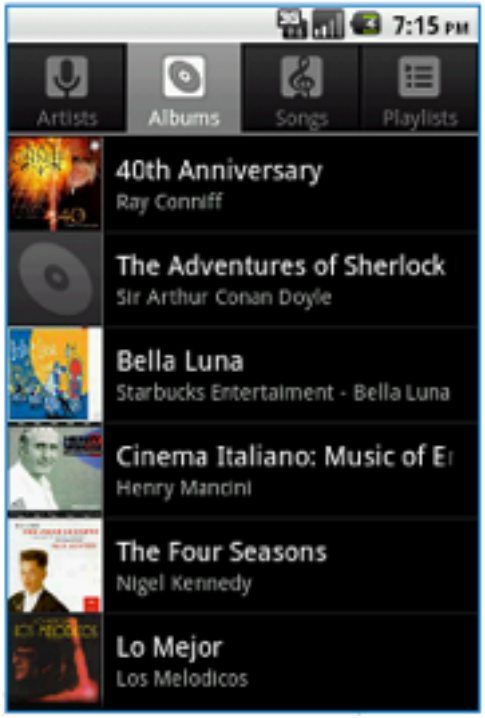
```
Intent intent = new Intent(Intent.ACTION_SENDTO,
Uri.parse("sms:5551234"));
intent.putExtra("sms_body", "are we playing golf next Saturday?");
startActivity(intent);
```

Intents

Example 1.6 : Showing Pictures (using extra attributes)

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

Intents



Example 1.7: Launching the Music player

Intent myIntent = **new Intent("android.intent.action.MUSIC_PLAYER");**
startActivity(myIntent);

Intents with getting results

Intents with getting results

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 with getting results

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 with getting results

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

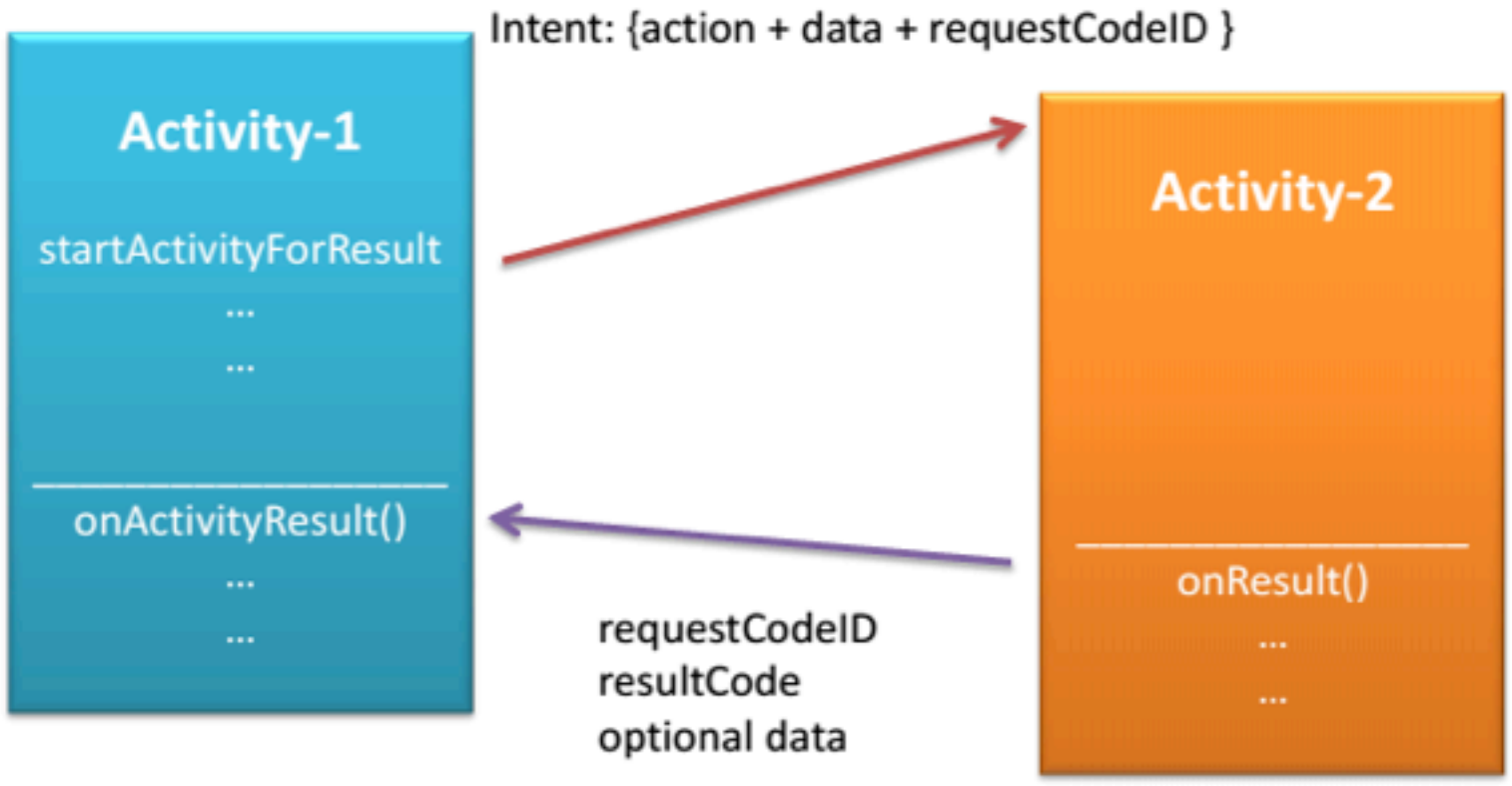
```
onActivityResult (int requestCodeID, int resultCode, Intent data)
```

along with the integer identifier it originally supplied.

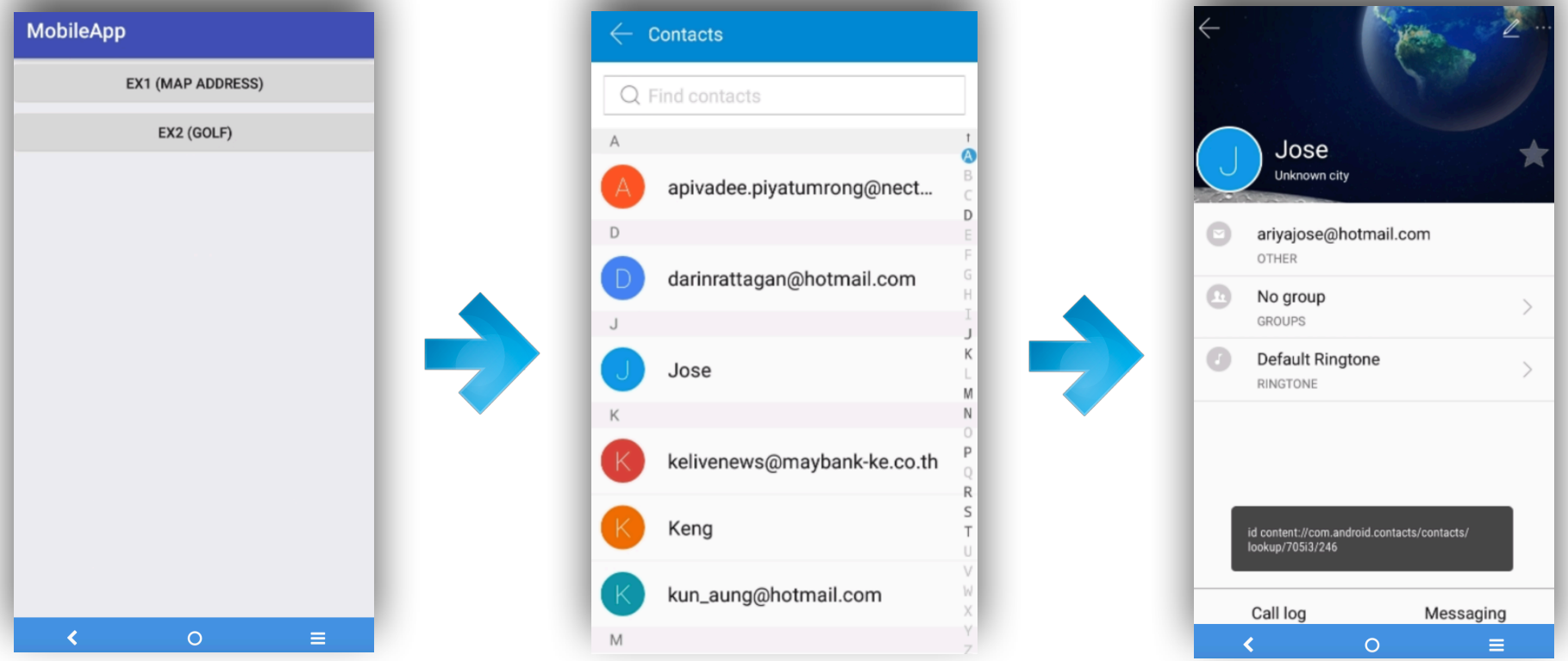
- If a child activity fails for any reason (such as crashing), the parent activity receive a result with the code **RESULT_CANCELED**.

Intents with getting results

Starting Activities and Getting Results



Example (Contact app)



1 2 3

1, 2

```
@Override
public void onClick(View v)
{
    try
    {

       Intent myIntent = new Intent(Intent.ACTION_PICK,
            android.provider.ContactsContract.Contacts.CONTENT_URI);

       startActivityForResult(myIntent,222);

    }
    catch(Exception e)
    {
        Log.d("Week4 (onClick) ",e.getMessage());
    }
}
```

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
   super.onActivityResult(requestCode, resultCode, data);
   try
   {
      switch(requestCode)
{
         case 222: {

            if(resultCode == Activity.RESULT_OK){

               String returnedData = data.getDataString();

Toast.makeText(getApplication(),"id " + returnedData,
Toast.LENGTH_LONG).show();

               //Call selected view.
               Intent myIntent2 = new Intent(Intent.ACTION_VIEW,
                        Uri.parse(returnedData));
               startActivity(myIntent2);
            }

            else{ }

            break;
         }
      }
   }
   catch(Exception e)
   {
      Log.d("Week4 (onActivityResult) “,e.getMessage());
```

What does this code do?

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

```
case 0:
   if(resultCode == Activity.RESULT_OK){

      String selectedItem = data.getDataString();

      Toast.makeText(getApplication(),"id " + selectedItem, Toast.LENGTH_LONG).show();

      //Call selected view.
      Intent myIntent3 = new Intent(Intent.ACTION_VIEW, Uri.parse(selectedItem));
      startActivity(myIntent3);
   }
   else{
      Log.d("Week4 onActivityResult ", "Selection Cancelled "+requestCode+ " " + resultCode);
   }
   break;
```

Taking a photo

AndroidManifest.xml

```
<manifest ... >
   <uses-feature android:name="android.hardware.camera"
                 android:required="true" />
   ...
</manifest>
```

OnClick

```
Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
    startActivityForResult(takePictureIntent,1);
}
```

Taking a photo

`onActivityResult`

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

    if (requestCode == 1 && resultCode == RESULT_OK) {

        Bundle extras = data.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");
        mImageView.setImageBitmap(imageBitmap);

    }

}
```