CPSC 475/575

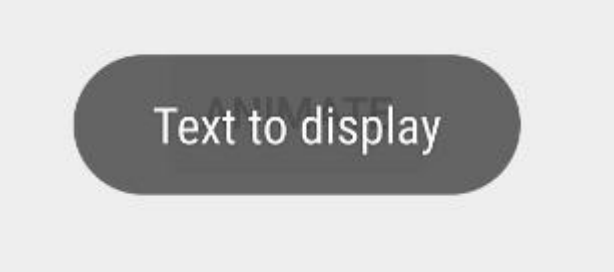
Invoking other applications using intents

Content adapted from
http://www.coreservlets.com/android-tutorial/
http://www.cs.utexas.edu/~scottm/cs378/schedule.htm
http://developer.android.com/guide/components/intents-filters.html
and other web resources

Odds and Ends

- Toasts are temporary messages that display information

```
Toast toast = Toast.makeText(this, "Text to display", Toast.LENGTH_SHORT);
toast.show();
```



Activation of Components

- 3 of the 4 core application components (activities, services, and broadcast receivers) are started via *intents*
- intents are a messaging system to activate components in the same application
- *and* to start one application from another

- We will just start Activities for now

startActivity, startActivityForResult and onActivityResult

startActivity

- startActivity is very simple, just begin the other app. You will use this a lot.

```
Intent myIntent = new Intent(this,SumActivity.class);
startActivity(myIntent);
```

- Does not return to your activity when ‘called’ app finishes

startActivityForResult

- startActivityForResult also starts the new activity, When it finishes the original activities onActivityResult is ‘called back’ by android

```
private void doScan() {
    //Ask a component to handle action  com.google.zxing.client.android.SCAN
    Intent intent = new Intent("com.google.zxing.client.android.SCAN");

    intent.putExtra("SCAN_MODE", "QR_CODE_MODE");
    startActivityForResult(intent, ID_DO_EXPLICIT_BARCODE);
}

protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch (requestCode) {
        case(ID_DO_EXPLICIT_BARCODE):
            doBarcode(resultCode,data);
            break;
    }
}
```

Intents

- Request something to happen
  (Explicit and Implicit, Next slide)

- Announce something has happened
  - android

```
android.bluetooth.a2dp.profile.action.CONNECTION_STATE_CHANGED
android.bluetooth.a2dp.profile.action.PLAYING_STATE_CHANGED
android.bluetooth.adapter.action.CONNECTION_STATE_CHANGED
android.bluetooth.adapter.action.DISCOVERY_FINISHED
android.bluetooth.adapter.action.DISCOVERY_STARTED
```

  - Your app

```
//explicit intent
Intent broadcastIntent = new Intent();
broadcastIntent.setAction(ResponseReceiver.ACTION_RESP);
broadcastIntent.addCategory(Intent.CATEGORY_DEFAULT);
broadcastIntent.putExtra(ResponseReceiver.MSG, "Just a dynamic message");

sendBroadcast(broadcastIntent);
```

Intents

- Request something to happen
  (Explicit and Implicit, Next slide)
- Announce something has happened
  - android

```
android.bluetooth.a2dp.profile.action.CONNECTION_STATE_CHANGED
android.bluetooth.a2dp.profile.action.PLAYING_STATE_CHANGED
android.bluetooth.adapter.action.CONNECTION_STATE_CHANGED
android.bluetooth.adapter.action.DISCOVERY_FINISHED
android.bluetooth.adapter.action.DISCOVERY_STARTED
```

  - Your app

```
//explicit intent
Intent broadcastIntent = new Intent();
broadcastIntent.setAction(ResponseReceiver.ACTION_RESP);
broadcastIntent.addCategory(Intent.CATEGORY_DEFAULT);
broadcastIntent.putExtra(ResponseReceiver.MSG, "Just a dynamic message");

sendBroadcast(broadcastIntent);
```

Used by Services, and Broadcast Receivers
a little of this now, more later

Intents and Activities

- Request something to happen
  - Explicit
    - I want YOU to do job (name exact class)

```
Intent myIntent = new Intent(this,SumActivity.class);
startActivity(myIntent);
```

  - Implicit
    - I want Someone who is capable of doing job (give general idea of what is required)

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_EMAIL, "kperkins@cnu.edu");
intent.putExtra(Intent.EXTRA_SUBJECT, "My Subject");
intent.putExtra(Intent.EXTRA_TEXT, "I am an email body.");
startActivity(Intent.createChooser(intent, "Send Email"));
```

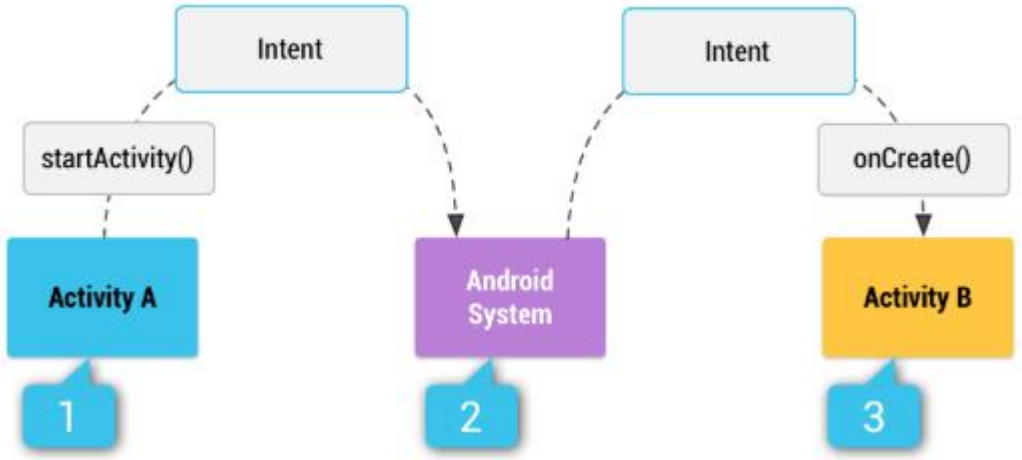
Intents that each component can handle are in manifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="scott.examples.lifeCycleTest"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="10" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".LifeCycleTestActivity"
            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=".NameGetter"
            android:label="@string/getName"/>
    </application>

</manifest>
```

Declare this as Activity to start when application started

Explicit
Select Exact Class

Intents – how they work
Explicit

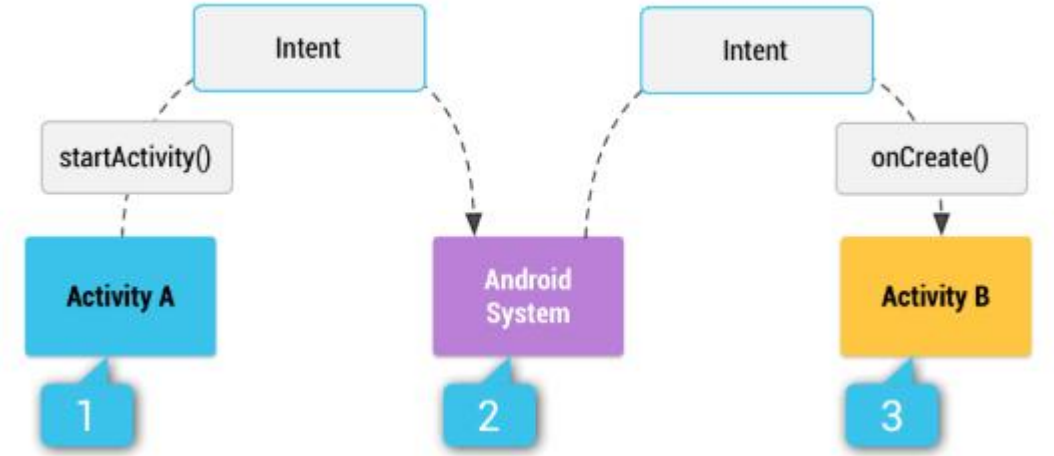


- Invoking a specific class (in your app or elsewhere on system)
  - Need fully qualified class name of component that should deal with Intent
- The Intent object is delivered to an instance of a SPECIFIC class by Android system. Note: this is how you start specific activities in your application or in other applications.

```
Intent myIntent = new Intent(this,SumActivity.class);
startActivity(myIntent);
```

Implicit
Provide general requirements
Let Android find class

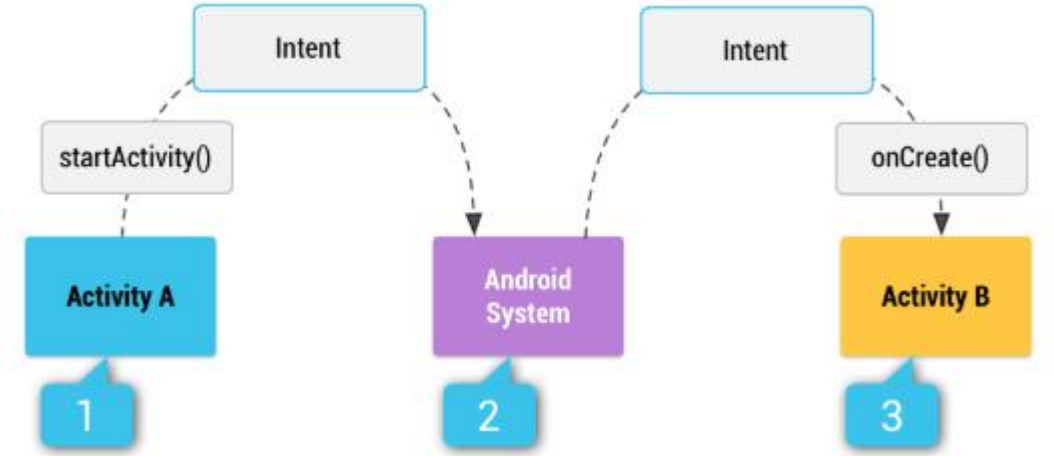
Intents – how they work
Implicit



- Let Android pick the component to start based on criteria you provide (you don't give it a class name).
- Android will choose a suitable component
- For instance; to start an activity that can take a picture

```
// create intent to take picture with camera
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
//start camera
startActivityForResult(intent, TAKE_PICTURE);
```

Intents – how they work
Implicit

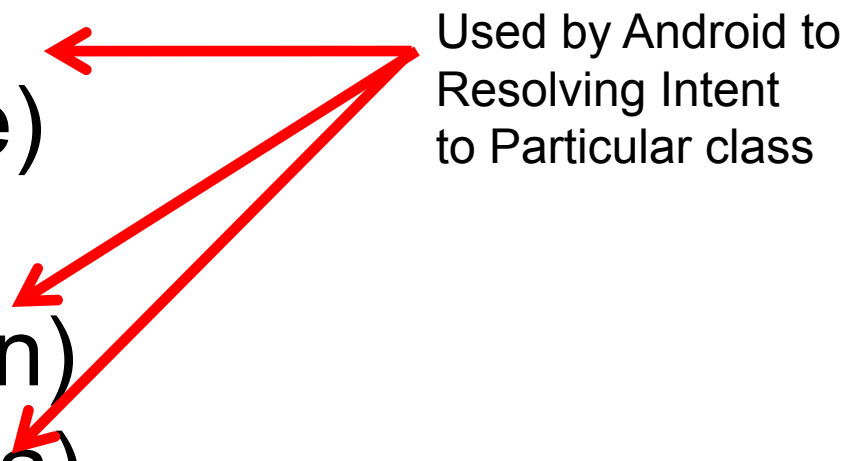


- Let Android pick the component to start based on criteria you provide (you don't give it a class name).
- Android will choose a suitable component
- Or, to start an activity that can send a message

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_EMAIL, "kperkins@cnu.edu");
intent.putExtra(Intent.EXTRA_SUBJECT, "My Subject");
intent.putExtra(Intent.EXTRA_TEXT, "I am an email body.");
startActivity(Intent.createChooser(intent, "Send Email"));
```

Intent Object Information

- component name (of desired component)
- action (to execute)
- data (to work on)
- category (of action)
- type (of intent data)
- extras (a Bundle with more data)
- flags (to help control how Intent is handled)



Intent *Action*

- Action acts like a method name
- determines what rest of data in Intent object is and how it is structured, especially the *data* and *extras*

Intent Action

Constant	Target component	Action
ACTION_CALL	activity	Initiate a phone call.
ACTION_EDIT	activity	Display data for the user to edit.
ACTION_MAIN	activity	Start up as the initial activity of a task, with no data input and no returned output
ACTION_SYNC	activity	Synchronize data on a server with data on the mobile device.
ACTION_BATTERY_LOW	broadcast receiver	A warning that the battery is low.
ACTION_HEADSET_PLUG	broadcast receiver	A headset has been plugged into the device, or unplugged from it.
ACTION_SCREEN_ON	broadcast receiver	The screen has been turned on.
ACTION_TIMEZONE_CHANGED	broadcast receiver	The setting for the time zone has changed.

Register You App for Common Actions

- Handle email?
- In manifest add following intent

```
<intent-filter>
    <action android:name="android.intent.action.SEND" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="text/plain" />
    <data android:mimeType="image/*" />
</intent-filter>
```

- see 4_IntentRegisterBogusEmailClient
  in 4_Explicit_implicit_Intentdemo

Create Your own Actions

- Register my apps custom action
- In manifest add following intent

```
<intent-filter>
    <action android:name="com.example.custom_intent.YOUR_ACTION" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

- To invoke from other app;

```
Intent myIntent = new Intent("com.example.custom_intent.YOUR_ACTION");
startActivity(myIntent);
```

- see 4_Explicit_implicit_Intentdemo

Passing Data from Class to Class via Bundles (see Appanatomy Lecture)

The Bundle Class: Details

- Putting data in a Bundle
  - putBoolean, putBooleanArray, putDouble, putDoubleArray, putString, putStringArray, putStringArrayList etc.
    - These all take keys and values as arguments. The keys must be Strings. The values must be of the standard types (int, double, etc.) or array of them.
- Retrieving data from a Bundle
  - getBoolean, getBooleanArray, getDouble, getDoubleArray, getString, getStringArray, getStringArrayList, etc.
    - These take keys (Strings) as arguments.

Option 1: Attaching Entire Bundle to Intent

- Idea
  - Make a Bundle, add it all at once to Intent.
    - Instantiate a Bundle, then use the Bundle's put*Blah* method (one such method for each standard type). Then, attach Bundle to Intent with Intent's putExtras method.
- Syntax

```
Bundle newActivityInfo = new Bundle();
newActivityInfo.putDouble("key1", someDouble);
newActivityInfo.putString("key2", someString);
…
yourIntent.putExtras(newActivityInfo);
```

Option 2: Adding One Piece of Data at a Time to Intent

- Idea
  - Add individual pieces of data to the Intent. No need to explicitly create and attach a Bundle.
    - You use the overloaded "putExtra" method. The first argument is the key (String), and the second argument is the value, which can be of any standard type. However, the code that retrieves the value later needs to know type.
- Syntax

```
yourIntent.putExtra("key1", someDouble);
yourIntent.putExtra("key2", someString);
…
```

  - Unlike putBlah for Bundle, these putExtra methods return the Intent, so you can chain calls
    - » yourIntent.putExtra(…).putExtra(…) … .putExtra(…);

Bundle Code Summary

Java (original Activity)

```
Intent activityIntent = new Intent(this, LoanCalculatorActivity.class);

//create a bunch of name, value pairs of data to pass
Bundle loanInfo = new Bundle();
loanInfo.putDouble("loanAmount", 80.3);
loanInfo.putDouble("annualInterestRateInPercent", 20);
loanInfo.putLong("loanPeriodInMonths", 39);
loanInfo.putString("currencySymbol", "$");

//place bundle into intent
activityIntent.putExtras(loanInfo);

//start the next activity
//which BTW is in this application
//because we did not fully qualify the name above
startActivity(activityIntent);
```

In the just started activity

```
Intent intent = getIntent();
Bundle loanInfo = intent.getExtras();
if (loanInfo != null) {
    //retreive all the data in the bundle
```

Intent Resolution

- How does the Android system determine what component should handle an Intent?
- explicit
  - Intent designates target component by name
  - typically used for inter application messaging and activity starting. You will use this a lot.

```
public void showLoanPayments1(View clickedButton) {
    Intent activityIntent = new Intent(this, LoanCalculatorActivity.class);
    startActivity(activityIntent);
}
```

Intent Resolution - Implicit

- component name is blank (unknown)
- typically used when starting component in another application
- Android system uses data from Intent (action, category, data) and tries to find / match best component for job
- Uses *Intent Filters*

Intent Filters

- Applications and components that can receive implicit Intents advertise what they can do via Intent Filters in manifest.
- components with no Intent Filters can only receive explicit Intents
  - typical of many activities
- activities, services, and broadcast receivers can have one or more intent filters

Intent Filters

- Android system should know what application can do without having to start the component
  - before runtime
  - exception is Broadcast Receivers registered dynamically; they create IntentFilter objects at runtime
- intent filters generally declared as element of applications Manifest.xml file

IntentFilter - Example

- filter declares action, category, and data
- If it skips one then that one is not part of the filter when looking for a matching intent

```
<intent-filter>
    <action android:name="android.intent.action.SEND" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="text/plain" />
    <data android:mimeType="image/*" />
</intent-filter>
```

IntentFilter - Example

- The Android system populates the application launcher via IntentFilters

```
<activity
    android:name="com.example.custom_intent.MainActivity"
    android:label="5_Custom_Intent" >
    <intent-filter>
        <action android:name="com.example.custom_intent.YOUR_ACTION" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

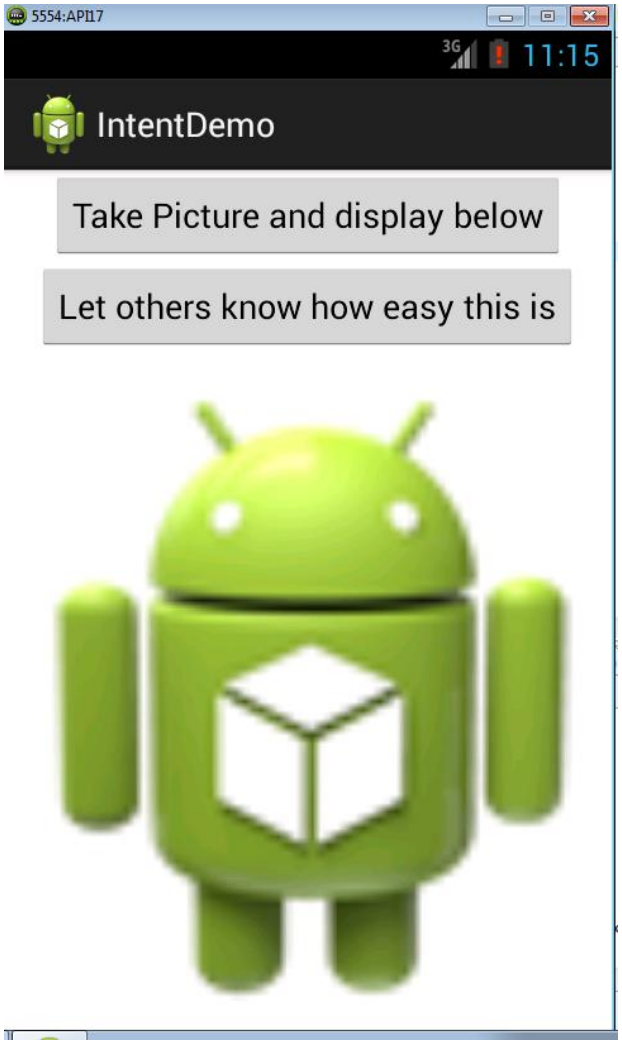
Summary

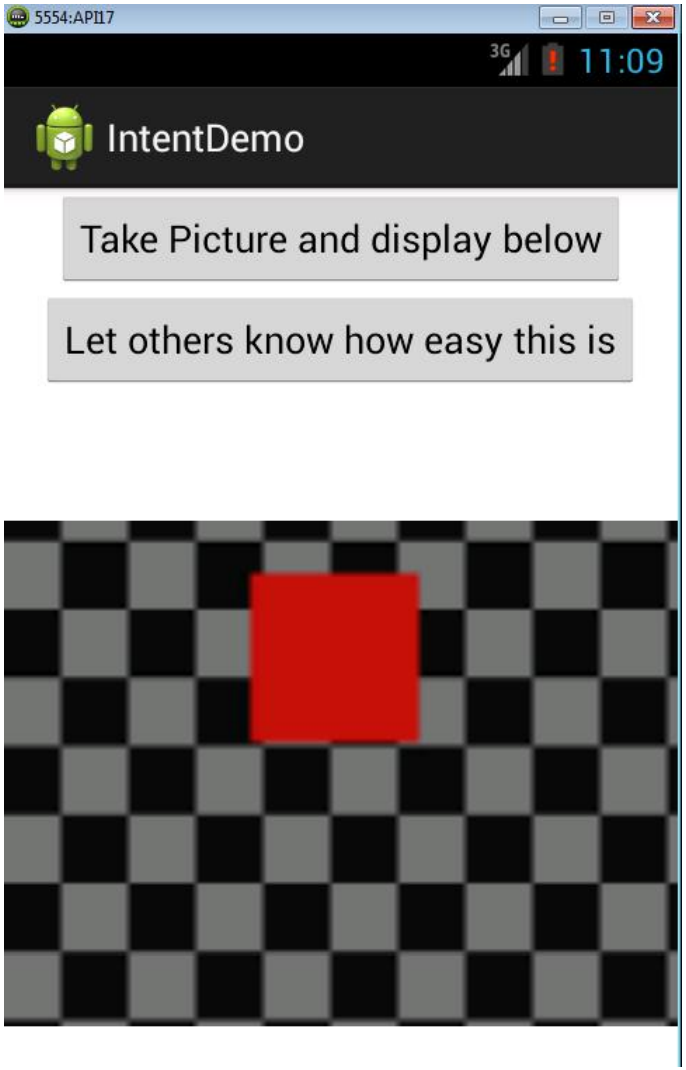
- Starting another activity and retreiving results from another activity
- Intents
  - (explicit)used to start your activities
  - (implicit)And to ask android to find an app to handle your needs

Example

- Use an Intent so app asks camera to take picture and displays the resulting picture
- important details:
  - permission to write and read (JellyBean) to and from SD card in manifest
  - getting file paths and names correct
  - reduce size of original image (do not hog memory)

IntentExample





Layout

- LinearLayout with
  - buttons
  - ImageView
- ImageView initially displays default Image
- button click results in call to launchCameraApp or LaunchCommsApp
  - android:onClick attribute set

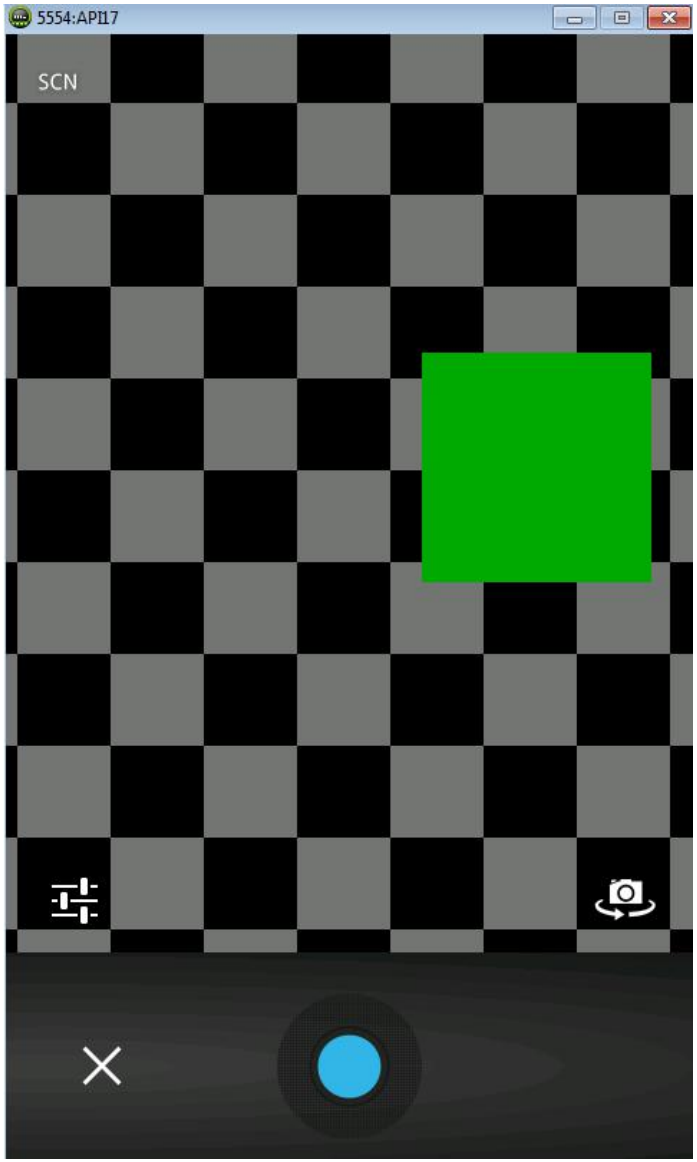
launchCameraApp in IntentActivity

Create an intent to take a picture using the camera and specify location and filename of photo so we can retrieve it easily

```
public void launchCameraApp(View v) {
    // create intent to take picture with camera and specify storage
    // location so we can easily get it
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    File file = new File(Environment.getExternalStorageDirectory(), "implicit.jpg");

    outputFileUri = Uri.fromFile(file);
    intent.putExtra(MediaStore.EXTRA_OUTPUT, outputFileUri);
    startActivityForResult(intent, TAKE_PICTURE);
}
```

Result

- Clicking button starts Camera Activity
- Parent activity will be stopped
  - recall Activity lifecycle, play well with others
- when picture taken return to CameraExample activity



onActivtyResult

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

        BitmapFactory.Options options = new BitmapFactory.Options();

        //You calculate this number usually, but in general and for simplicity 4 works
        //this meansreduce image size to 1/4 original
        options.inSampleSize = 4;

        //magic string, but useful for an example
        //this tells bitmapfactory where to find saved file
        String file = Environment.getExternalStorageDirectory() + "/test.jpg";
        Bitmap bitmap = BitmapFactory.decodeFile(file, options);

        //set the imageview to our bitmap
        image.setImageBitmap(bitmap);
    }
}
```

Activity Registers for action and category intent

```
<activity
    android:name="com.example.bogusemailclient.BogusActivity"
    android:label="@string/app name" >
    <intent-filter>
        <action android:name="android.intent.action.SEND" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="text/plain" />
        <data android:mimeType="image/*" />
    </intent-filter>
</activity>
```

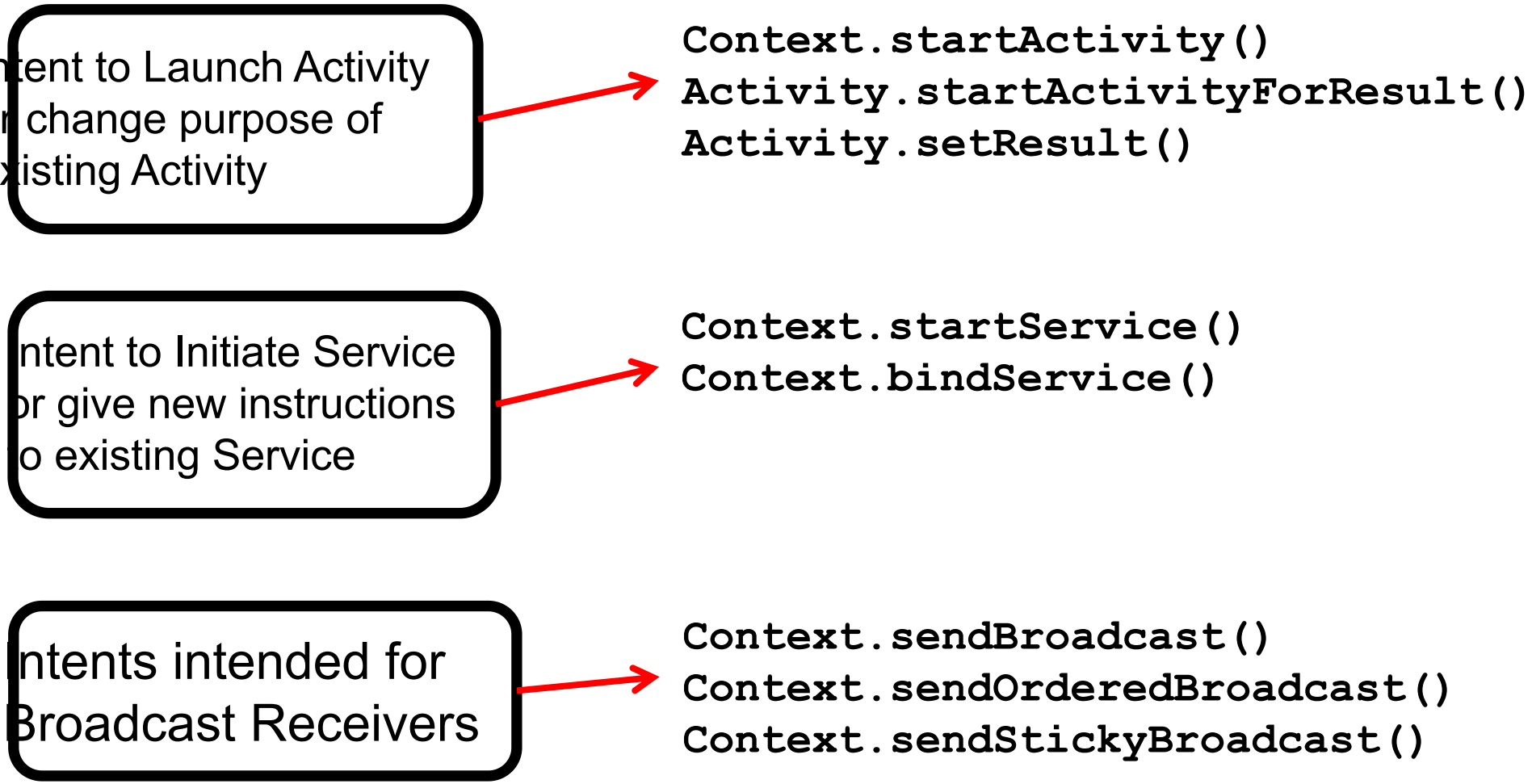
Activity Registers for action and category intent

```
<activity
    android:name="com.example.bogusemailclient.BogusActivity"
    android:label="@string/app name" >
    <intent-filter>
        <action android:name="android.intent.action.SEND" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="text/plain" />
        <data android:mimeType="image/*" />
    </intent-filter>
</activity>
```

Activity Registers for action and category intent

```
<activity
    android:name="com.example.bogusemailclient.BogusActivity"
    android:label="@string/app name" >
    <intent-filter>
        <action android:name="android.intent.action.SEND" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="text/plain" />
        <data android:mimeType="image/*" />
    </intent-filter>
</activity>
```

Intents and App Components



The Android System finds the right application component to respond to intents, instantiating them if necessary.

Intent Info - *Data*

- URI (uniform resource identifier) identifies a resource to work on
  - A general way to work with locations
  - for content on device a content provider and identifying information, for example an audio file or image or contact
- MIME (Multipurpose Internet Mail Extension, now internet media type) initially for email types, but extended to describe type information in

Intent Class and Objects

- android.content.Intent
- passive data structure
  - description of action to performed or if created by a broadcast, a description of something that has happened and is being announced to broadcast receivers
- Intent objects carry information, but do not perform any actions themselves

Intent Info - *Category*

- String with more information on what kind of component should handle Intent

Constant	Meaning
`CATEGORY_BROWSABLE`	The target activity can be safely invoked by the browser to display data referenced by a link — for example, an image or an e-mail message.
`CATEGORY_GADGET`	The activity can be embedded inside of another activity that hosts gadgets.
`CATEGORY_HOME`	The activity displays the home screen, the first screen the user sees when the device is turned on or when the *Home* button is pressed.
`CATEGORY_LAUNCHER`	The activity can be the initial activity of a task and is listed in the top-level application launcher.
`CATEGORY_PREFERENCE`	The target activity is a preference panel.