

# CS378 - Mobile Computing

Intents

# Intents

- Allow us to use applications and components that are part of Android System
  - start activities
  - start services
  - deliver broadcasts
- and allow other applications to use the components of the applications we create
- Examples of Google applications:  
<http://developer.android.com/guide/appendix/g-app-intents.html>

# Intents

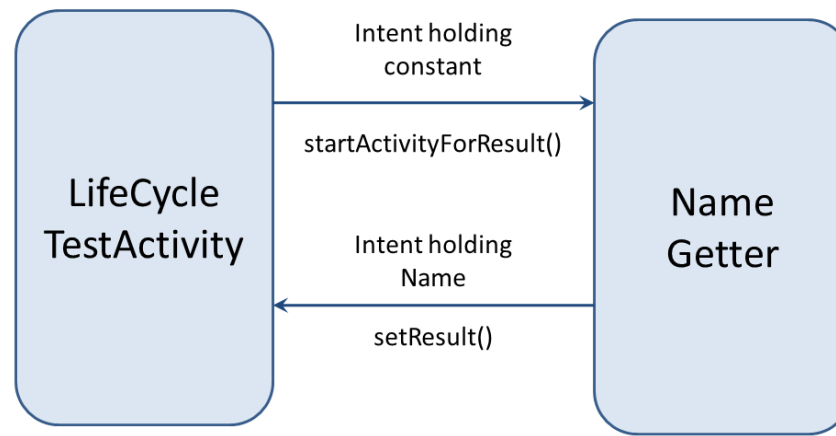
- "An intent is an abstract description of an operation to be performed"
- consist of
  - Action (what to do, example visit a web page)
  - Data (to perform operation on, example the url of the web page)
- use via startActivity, startActivityForResult, startService, bindService

# Four Primary Application Components

- Activity
  - single screen with a user interface, app may have several activities, subclass of Activity
- Service
  - Application component that performs long-running operations in background with no UI
- Content Providers
  - a bridge between applications to share data
- Broadcast Receivers
  - component that responds to system wide announcements

# 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



# AndroidManifest.xml

- Recall the manifest is part of the application project.
- The manifest contains important data about the application that is required by the Android system before the system will run any of the application's code
  - common error: Activity in application that is not included in manifest
  - runtime error when application tries to start Activity not declared in manifest

# AndroidManifest.xml Purpose

- contains Java package name of application - unique id for application
- describes components of application: activities, services, broadcast receivers, content providers and *intent messages each component can handle*
- declares permissions requested by application
- minimum required API level
- libraries application to link to

# AndroidManifest.xml - Launcher Intent

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="scott.examples.lifeCycleTest"
4     android:versionCode="1"
5     android:versionName="1.0" >
6
7     <uses-sdk android:minSdkVersion="10" />
8
9     <application
10         android:icon="@drawable/ic_launcher"
11         android:label="@string/app_name" >
12         <activity
13             android:name=".LifeCycleTestActivity"
14             android:label="@string/app_name" >
15             <intent-filter>
16                 <action android:name="android.intent.action.MAIN" />
17                 <category android:name="android.intent.category.LAUNCHER" />
18             </intent-filter>
19         </activity>
20         <activity
21             android:name=".NameGetter"
22             android:label="@string/getName"/>
23     </application>
24
25 </manifest>
```

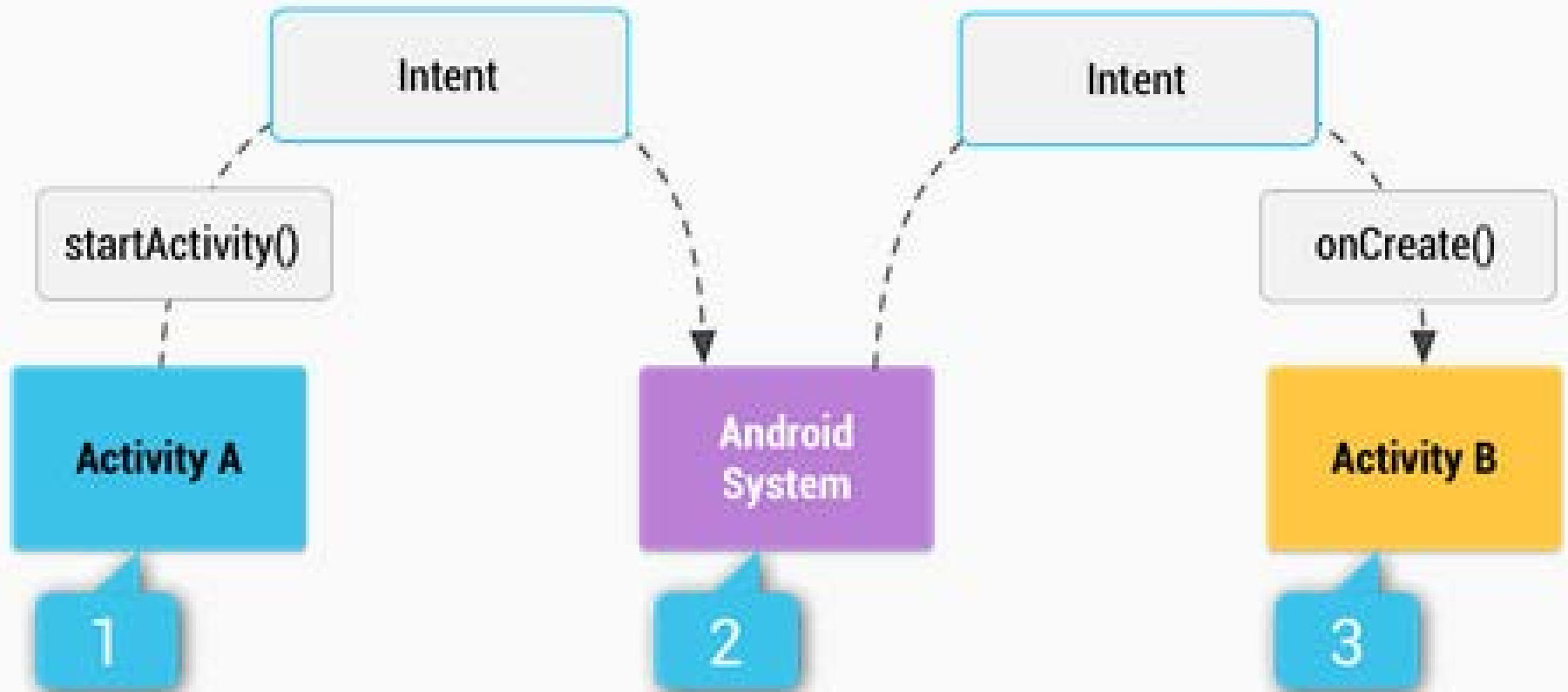
Declare this as Activity  
to start when application  
started



# Types of Intents

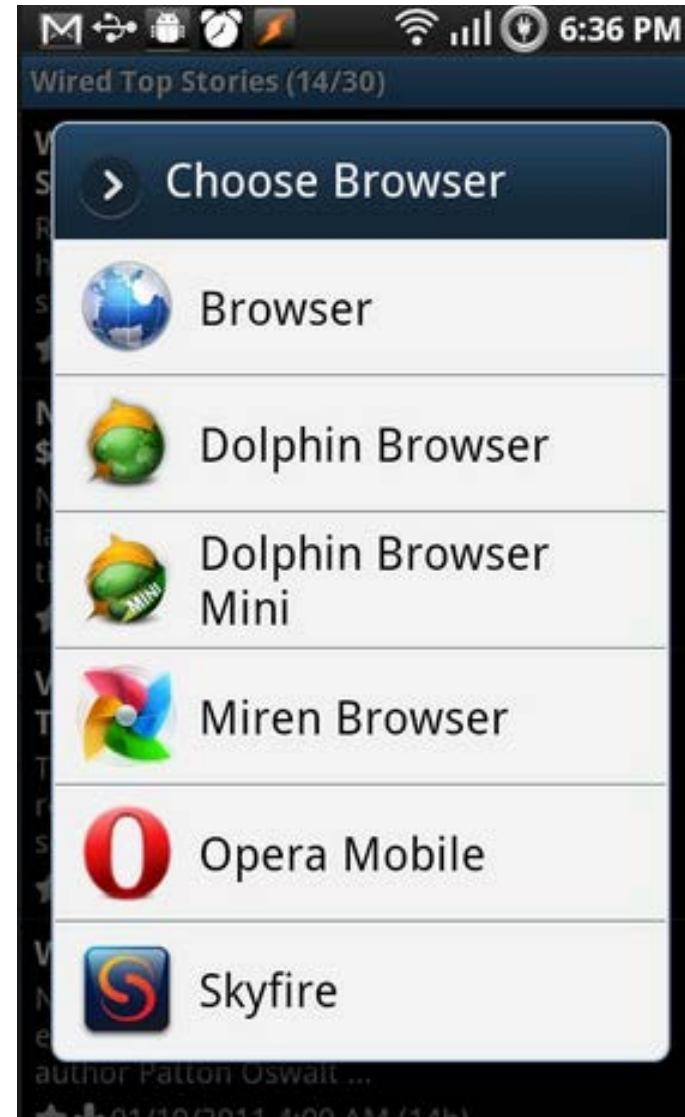
- <http://developer.android.com/reference/android/content/Intent.html>
- Categories include:
- Action
  - e.g. VIEW, EDIT, CALL, DIAL, SEARCH
- Broadcast Action
  - e.g. TIMEZONE\_CHANGED,  
POWER\_CONNECTED

# Implicit Intents



# Multiple Apps To Handle Intent

- Intent class contains constants for Intents
- Applications and activities list intents they can handle in manifest
- If multiple available asked to choose
- `android.intent.action.WEB_SEARCH`



# 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

# Intents and App Components

Intent to Launch Activity  
or change purpose of  
existing Activity

`Context.startActivity()`  
`Activity.startActivityForResult()`  
`Activity.setResult()`

Intent to Initiate Service  
or give new instructions  
to existing Service

`Context.startService()`  
`Context.bindService()`

Intents intended for  
Broadcast Receivers

`Context.sendBroadcast()`  
`Context.sendOrderedBroadcast()`  
`Context.sendStickyBroadcast()`

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

# Intent Object Information

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

# Intent Object Info

- data for the component that receives the intent
  - action to take
  - data to act on
- data for the Android system
  - category of component to handle intent (activity, service, broadcast receiver)
  - instructions on how to launch component if necessary

# Intent Constructors

## 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)`

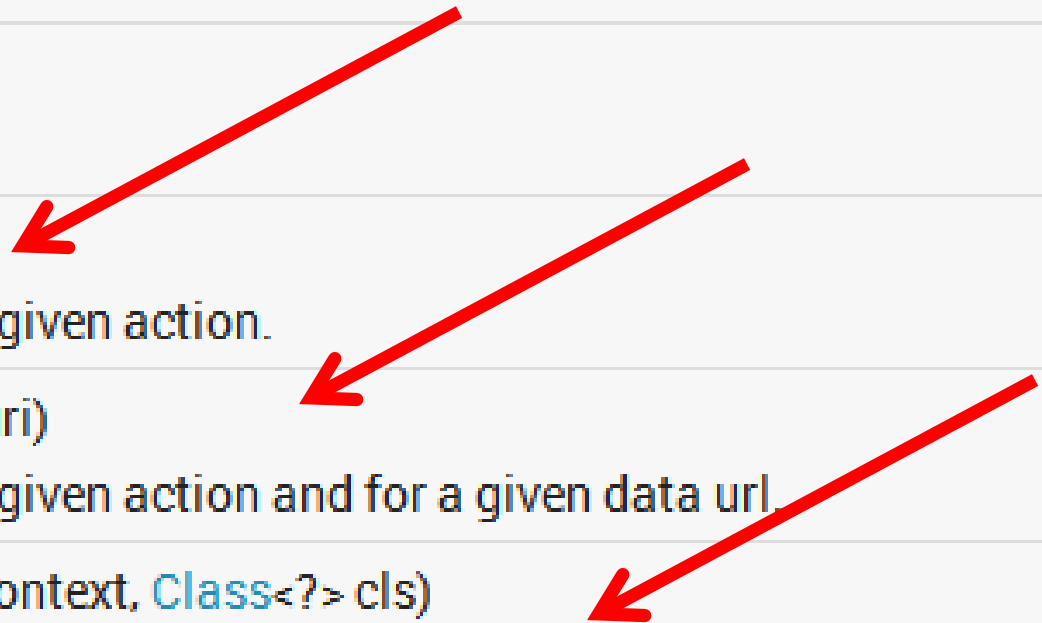
Create an intent with a given action and for a given data url.

`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.

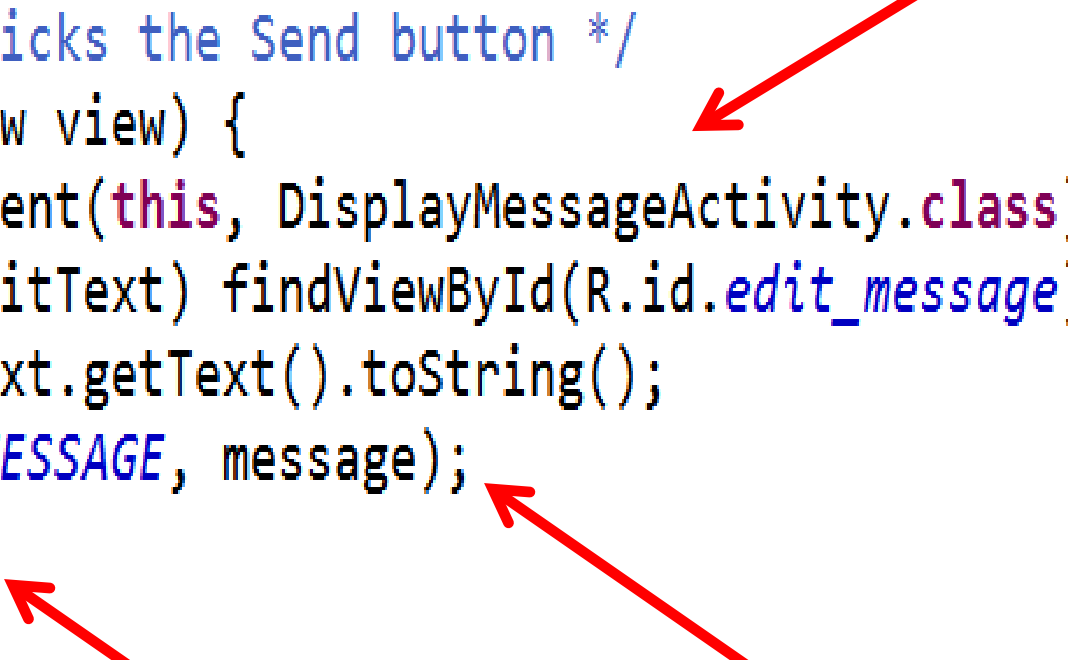


# Intent Info - *Component*

- Component name that should deal with Intent
- fully qualified class name of component and
- the package name set in the manifest file of the application where the component resides
- optional!
  - if not provided Android system resolves suitable target
- name is set by `setComponent()`, `setClass()`, or `setClassName()`

# From MyFirstActivity

```
/** Called when the user clicks the Send button */  
public void sendMessage(View view) {  
    Intent intent = new Intent(this, DisplayMessageActivity.class);  
    EditText editText = (EditText) findViewById(R.id.edit_message);  
    String message = editText.getText().toString();  
    intent.putExtra(EXTRA_MESSAGE, message);  
    startActivity(intent);  
}
```



```
public final static String EXTRA_MESSAGE  
    = "scottm.utexas.myfirstapp.MESSAGE";
```

## Intent Info - *Action Name*

- Action desired (or for broadcast intents, the action / event that took place)
- Many actions defined in Intent class
- Other actions defined through the API
  - example, MediaStore class declares ACTION\_IMAGE\_CAPTURE and ACTION\_VIDEO\_CAPTURE
- You can define your own Intent Action names so other applications can activate the components in your application

# Intent *Action Name*

- 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*
- `setAction()` and `getAction()` methods from Intent class

# Intent Action

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

# Intent Info - *Data*

- URI (uniform resource identifier) of data to work with / on
  - 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 general about data / content
  - `image/png` or `audio/mpeg`

# Intent Info - *Category*

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

Constant	Meaning
<code>CATEGORY_BROWSABLE</code>	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.
<code>CATEGORY_GADGET</code>	The activity can be embedded inside of another activity that hosts gadgets.
<code>CATEGORY_HOME</code>	The activity displays the home screen, the first screen the user sees when the device is turned on or when the <i>Home</i> button is pressed.
<code>CATEGORY_LAUNCHER</code>	The activity can be the initial activity of a task and is listed in the top-level application launcher.
<code>CATEGORY_PREFERENCE</code>	The target activity is a preference panel.

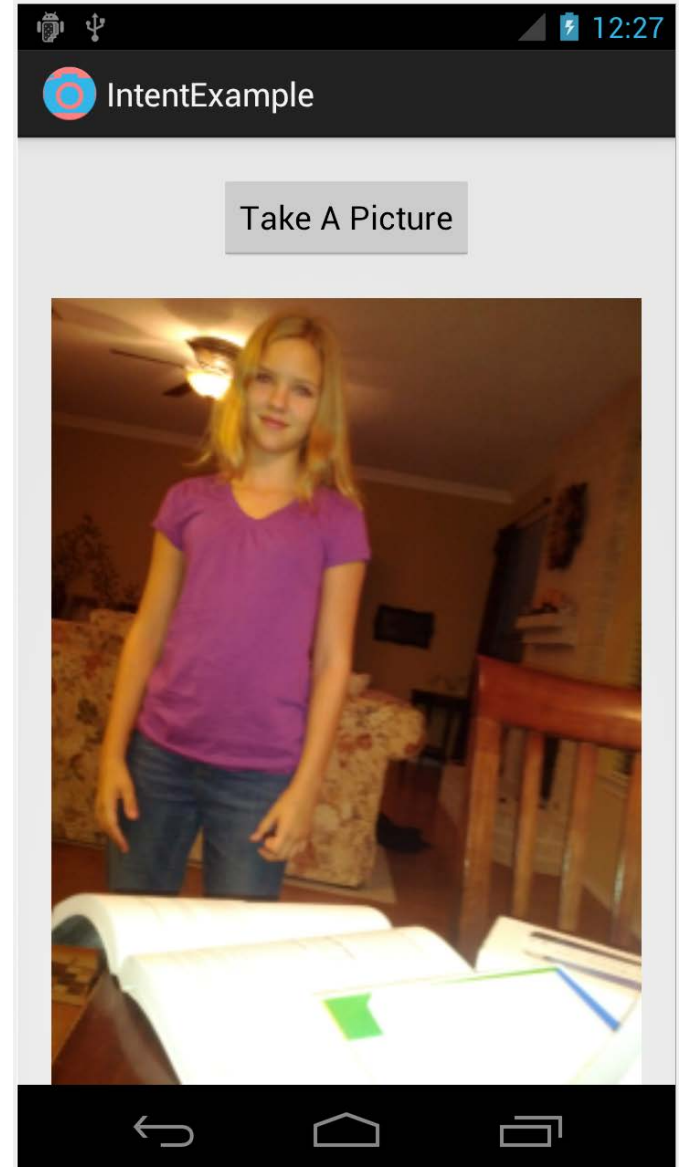
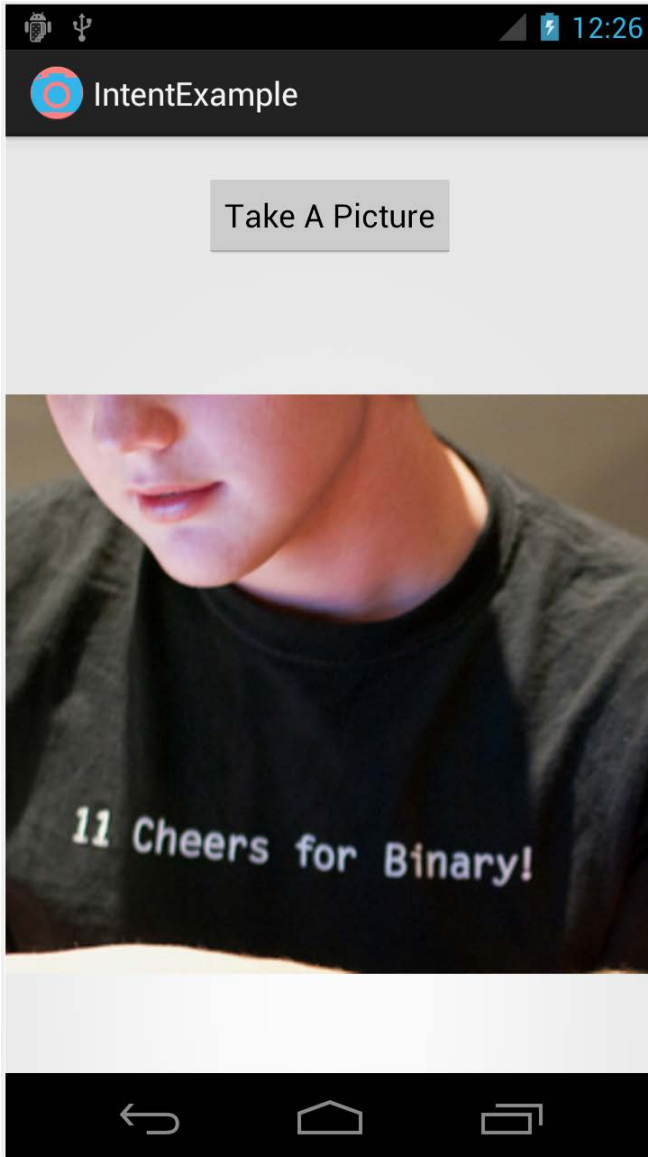
# Intent - *Extras*

- A *Bundle* (like a map / dictionary, key-value pairs) of additional information to be given to the component handling the Intent
- Some Action will have specified extras
  - ACTION\_TIMEZONE\_CHANGED will have an extra with key of "time-zone"  
(documentation is your friend)
  - Intent method has put methods or put a whole Bundle

# 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
  - getting file names correct
  - reduce size of original image

# IntentExample



# Layout

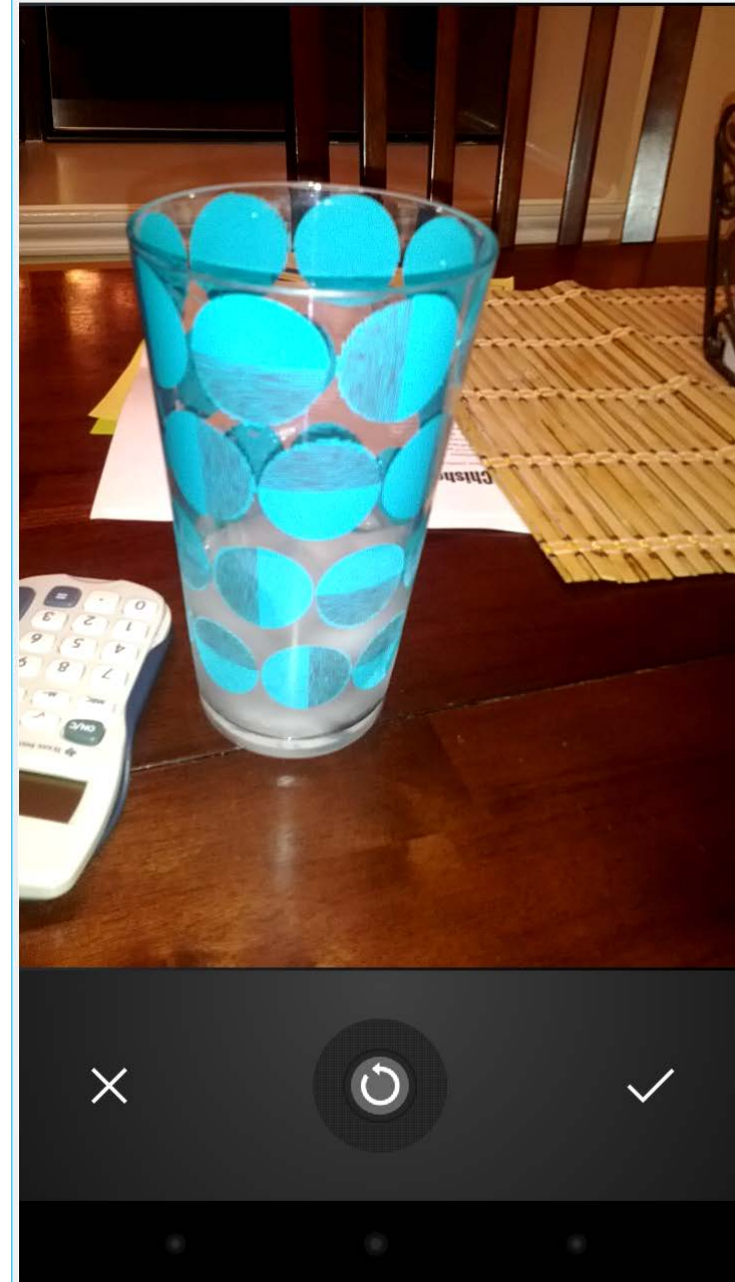
- LinearLayout with
  - button
  - ImageView
- ImageView initially displays default Image
- button click results in call to takePhoto
  - android:onClick attribute set

# takePhoto in IntentExample Activity

```
public void takePhoto(View v) {  
    // create directory if necessary  
    File photoDir  
        = new File(Environment.getExternalStorageDirectory()  
            + "/intentExamplePhotos/");  
  
    if(photoDir.mkdirs())  
        Log.d(TAG, "mkdirs returned true: " + photoDir);  
    else  
        Log.d(TAG, "mkdirs returned false: " + photoDir);  
  
    // create Intent to take picture via camera and specify location  
    // to store image so we can retrieve easily  
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);  
    File file = new File(fileName);  
    outputFileUri = Uri.fromFile(file);  
    intent.putExtra(MediaStore.EXTRA_OUTPUT, outputFileUri);  
    startActivityForResult(intent, TAKE_PICTURE);  
}
```

# Result

- Clicking button starts Camera Activity
- IntentExample stops
  - recall Activity lifecycle, play well with others
- when picture taken return to IntentExample activity



# onActivityResult

- when camera app checks Android system will call this method (callback)
- look at result and take appropriate action
- verify our requested action was completed

# onActivityResult

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

    ImageView img = (ImageView)this.findViewById(R.id.imageView1);

    if (requestCode == TAKE_PICTURE && resultCode == RESULT_OK){
        // change picture in ImageView to image just taken

        // reduce size of image
        BitmapFactory.Options options = new BitmapFactory.Options();
        options.inSampleSize = 4;
        Bitmap bmp = BitmapFactory.decodeFile(fileName, options);
        img.setImageBitmap(bmp);

        Toast.makeText(this, "Photo saved to: "
            + outputFileUri.toString(), Toast.LENGTH_LONG).show();

        Log.d(TAG, "Photo saved to: " + outputFileUri.toString());
    }
}
```

# onActivityResult

```
else if(resultCode == RESULT_CANCELED) {  
    Bitmap onPictureImage  
        = BitmapFactory.decodeResource(getResources(),  
            R.drawable.no_picture);  
    img.setImageBitmap(onPictureImage);  
}
```

```
Log.d(TAG, "request code: " + requestCode);  
Log.d(TAG, "result code: " + resultCode);
```



No Picture Taken

# 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
  - recall, LifecycleTest

```
public void getName(View v) {  
    Intent intent = new Intent(this, NameGetter.class);  
    startActivityForResult(intent, GET_NAME);  
}
```

# 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
- *Intent Filters*

# Intent Filters

- Applications and components that can receive implicit Intents advertise what they can do via Intent Filters
- 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 AndroidManifest.xml file

# IntentFilter - Example

- filter declares action, category, and data

```
<activity android:name="TitleEditor"
    android:label="@string/title_edit_title"
    android:theme="@android:style/Theme.Dialog">
    <intent-filter android:label="@string/resolve_title">
        <action android:name="com.android.notepad.action.EDIT_TITLE" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.ALTERNATIVE" />
        <category android:name="android.intent.category.SELECTED_ALTERNATIVE" />
        <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
    </intent-filter>
</activity>
```

# IntentFilter - Example

- The Android system populates the application launcher via IntentFilters

```
<activity
    android:name=".IntentExample"
    android:label="@string/title_activity_intent_example" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
...

```