

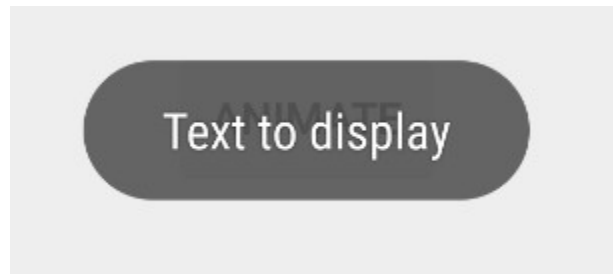
CPSC 475/575

Invoking other applications
using intents

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 activity. You will use this a lot.

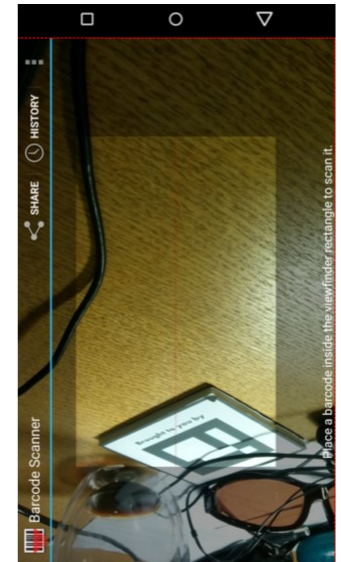
```
Intent myIntent = new Intent(this, SumActivity.class);  
startActivity(myIntent);
```

- Does not return to your activity when 'called' app finishes

startActivityResult

- startActivityResult also starts a 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");  
    startActivityResult(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)

- Or...announce that 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)

- Or...announce that something has happened

– Used by Services, and Broadcast Receivers

– a little of this now, more later

– Your app

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

//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);
```


Explicit and Implicit Intents

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

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

– Implicit

- I want Something that is capable of doing a 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

```
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 </manifest>
```

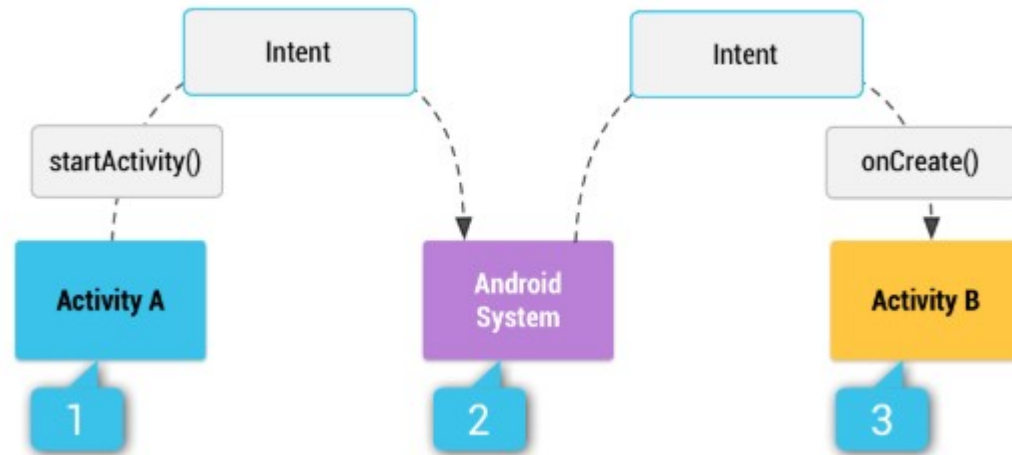
Declare this as Activity
to start when application
started



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);
```

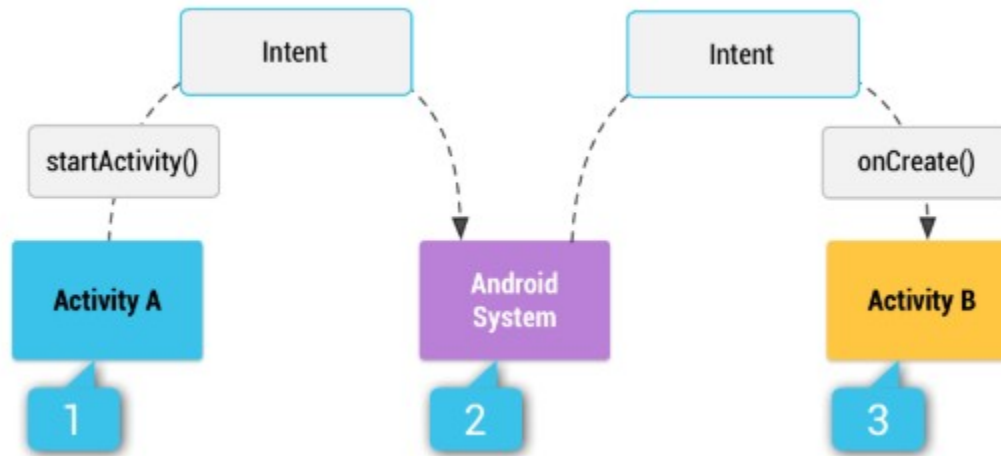
Implicit

Provide general
requirements

Let Android find class

Intents – how they work

Implicit

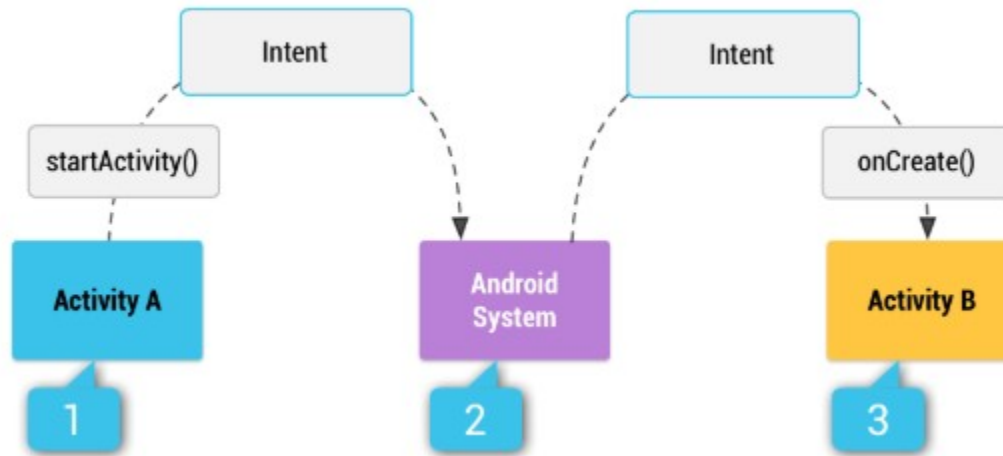


- Let Android pick the component to start based on criteria you provide (you don't give it a class name).
- Android will choose a suitable component
- For instance; to start an activity that can take a picture

```
// 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

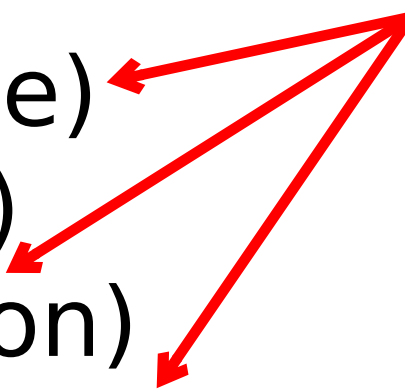
Implicit



- 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)
- Used by Android to Resolving Intent to Particular 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.

Register Your 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 *putBlah* 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 like so

```
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);
```

Java (new Activity)

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

In the just started activity



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

- The Android system populates the application launcher via IntentFilters

From 4_CustomIntent_and_BogusEmail_App, register custom action

```
<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>
```

From 4_Explicit_Implicit_IntentDemo, launch above via custom action

```
Intent myIntent = new Intent( action: "com.example.custom_intent.YOUR_ACTION");
if (myIntent.resolveActivity(getPackageManager()) == null) {
    // Error occurred while creating the File
    Toast.makeText( context: this, text: "INSTALL 4_CustomIntent_and_BogusEmail_App FIRST", Toast.LENGTH_SHORT).show();
    return;
}
startActivity(myIntent);
```

Summary

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