# Upcoming Assignments

- Readings: Chapter 5 by today
- Horizontal Prototype due Friday, January 22
  - Be prepared to demo them in class
- Lab Quiz Monday during lecture time (2:10-3pm)
  - Have a development environment in 256 or 255
  - Be familiar with the application lifecycle and Toast
- Lab 2 due today, survey
- Lab 3 due Wednesday, January 27

# How-to's

- Write your gmail account on form
- I will add you as a collaborator
- Create a new post
  - Demo
    - Create page
    - Add item to How-to list

# Google I/O

- Consider attending Google I/O in San Francisco May 19-20
- $100 for academic registration
- Academic registrations are limited
- Android tracks
- Last year gave away MyTouch Androids
  - Don’t count on anything this year

# Intents

- Allows communication between loosely-connected components
- Allows for late run-time binding of components
- Explicit

```
Intent myIntent = new Intent(AdventDevos.this, Devo.class);
myIntent.putExtra("ButtonNum", ""+index);
startActivity(myIntent);
//finish();  //removes this Activity from the stack
```

- Implicit

```
Intent i = new Intent(Intent.ACTION_VIEW,
      Uri.parse("http://www.biblegateway.com/passage/?search="+
      passage +"&version=NIV"));
startActivity(i);
```

# Other Native Android Actions

- ACTION_ANSWER – handle incoming call
- ACTION_DIAL – bring up dialer with phone #
- ACTION_PICK – pick item (e.g. from contacts)
- ACTION_INSERT – add item (e.g. to contacts)
- ACTION_SENDTO – send message to contact
- ACTION_WEB_SEARCH – search web

# Sub-Activities

- Activities are independent
- However, sometimes we want to start an activity that gives us something back (e.g. select a contact and return the result)
- Use

  startActivityForResult(Intent i, int id)

  instead of

  startActivity(Intent)

# Capturing Intent Return Results

```
class ParentActivity extends Activity {
   private static final int SUB_CODE = 34;
…
        Intent intent = new Intent(…);
        startActivityForResult(intent, SUB_CODE);
…
   @Override
   public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == SUB_CODE)
            if (resultCode == Activity.RESULT_OK) {
                Uri returnedUri = data.getData();
                String returnedString = data.getStringExtra(SOME_CONSTANT,"");
                …
            }
            if (resultCode == Activity.RESULT_CANCELED) { … }
   }
};
```

# Capturing Intent Return Results

```
class SubActivity extends Activity {
…
    if (/* everything went fine */) {
        Uri data = Uri.parse(“content://someuri/”);
        Intent result = new Intent(null,data);
        result.putStringExtra(SOME_CONSTANT, “This is some data”);
        setResult(RESULT_OK, result);
        finish();
    }
…
    if (/* everything did not go fine or the user did not complete the action */) {
        setResult(RESULT_CANCELED, null);
        finish();
    }
…
};
```

# Using a Native App Action

```
public class MyActivity extends Activity {
    //from http://developer.android.com/reference/android/app/Activity.html
    ...
    static final int PICK_CONTACT_REQUEST = 0;
    protected boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
            // When the user center presses, let them pick a contact.
            startActivityForResult(
                new Intent(Intent.ACTION_PICK, new Uri("content://contacts")),
                                                        PICK_CONTACT_REQUEST);
           return true;
        }
        return false;
    }
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == PICK_CONTACT_REQUEST) {
            if (resultCode == RESULT_OK) {
                // A contact was picked.  Here we will just display it to the user.
                startActivity(new Intent(Intent.ACTION_VIEW, data));
            }
        }
```

# Broadcasts and Broadcast Receivers

- So far we have used Intents to start Activities
- Intents can also be used to send messages anonymously between components
- Messages are sent with **sendBroadcast()**
- Messages are received by extending the **BroadcastReceiver** class

# Sending a Broadcast

```
//…
        public static final String MAP_ADDED =
            “com.simexusa.cm.MAP_ADDED”;
//…
        Intent intent = new Intent(MAP_ADDED);
        intent.putStringExtra(“mapname”, “Cal Poly”);
        sendBroadcast(intent);
//…
```

Typically like a package name to keep it unique

# Receiving a Broadcast

```
public class MapBroadcastReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        Uri data = intent.getData();
        String name = data.getStringExtra("mapname");
        //do something
        context.startActivity(…);
    }
};
```

Must complete in <5 seconds

- Broadcast Receivers are started automatically – you don't have to try to keep an Activity running

# Registering a BroadcastReceiver

- Statically in ApplicationManifest.xml

```
<receiver android:name=".MapBroadcastReceiver">
  <intent-filter>
    <action android:name="com.simexusa.cm.MAP_ADDED">
  </intent-filter>
</receiver>
```

or dynamically in code (e.g. if only needed while visible)

```
IntentFilter filter = new IntentFilter(MAP_ADDED);
MapBroadcastReceiver mbr = new MapBroadcastReceiver();
registerReceiver(mbr, filter);
…
unregisterReceiver(mbr);
```

in onRestart() ?

in onPause() ?

# Native Broadcasts

- ACTION_CAMERA_BUTTON
- ACTION_TIMEZONE_CHANGED
- ACTION_BOOT_COMPLETED
  - requires RECEIVE_BOOT_COMPLETED permission

# Intent Filters

- Intent filters register application components with Android
- Intent filter tags:
  - action – unique identifier of action being serviced
  - category – circumstances when action should be serviced (e.g. ALTERNATIVE, DEFAULT, LAUNCHER)
  - data – type of data that intent can handle
    - Ex. URI = content://com.example.project:200/folder/subfolder/etc
- Components that can handle implicit intents (one's that are not explicitly called by name), must declare category DEFAULT or LAUNCHER

```
<activity android:name="NotesList" android:label="@string/title_notes_list">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <action android:name="android.intent.action.EDIT" />
        <action android:name="android.intent.action.PICK" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.GET_CONTENT" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
    </intent-filter>
</activity>

<activity android:name="NoteEditor"
          android:theme="@android:style/Theme.Light"
          android:label="@string/title_note" >
    <intent-filter android:label="@string/resolve_edit">
        <action android:name="android.intent.action.VIEW" />
        <action android:name="android.intent.action.EDIT" />
        <action android:name="com.android.notepad.action.EDIT_NOTE" />
        <category android:name="android.intent.category.DEFAULT" />
```

# Intent Resolution

- See p. 123 in text