**Android Tutorial Notes**

**Meeting Sunday 10/2/11**

**Nora Ng-Quinn**

**Disclaimer:** This is not a full guide to using Android, but simply a basic summary. Most of my information was taken from <http://developer.android.com/guide/index.html> so please check it out for more detailed information.

**Basic Building Blocks**

**Activities**   
A single screen with a user interface.   
Example - An email application with one activity to show a list of emails, another to compose the mail, another to read the email.   
  
**Services**  
Runs in the background and performs long-running operations. No user interface. Example - itunes that plays music in the background while you are surfing Firefox.  
  
**Content Providers**  
Manages a shared set of application data.   
Example - maintains the user’s contact list. Applications may read from or write to the contact list through the content provider.  
  
**Broadcast Receivers**  
A component that responds to system-wide broadcast announcements.  
Example - Announcement that battery is low, that picture is captured, download complete  
  
**How Android is Different**

* Activities can start other activities. No base activity to start all activities.
* Android applications don’t have a single entry point (no main() function).
* Usually applications cannot directly activate another activity’s component. In Android, you can deliver a message to the system through an *intent* to start another activity’s component. The system them activates it for you.
* Still programs in java, but also works in conjunction with XML files, and Android components

**Intents**  
The messengers that request an action from other components. Can activate an activity, service, or broadcast receiver. Intent holds a request to perform that is sent to other components. For activities there sometimes is a return intent that may hold your result. Content Providers not activated by intents, instead activated by requests from content resolver.  
Example - (Intent to Activity) An intent to let the user pick a personal contact, return intent with URI pointing to chosen contact. (Intent to Broadcast receiver) intent holding message “battery is low”.

**Setting Up**

**Creating an Android Class**

1. Click File>New>New Android Project. I menu should show up.
2. Input Project Name (for Eclipse), Application Name (name that will show up on phone), package name (for eclipse package under Eclipse project), and Create Activity (the name of the first activity to be initialized, like main()).

**Emulator**

The Android simulator that runs your project on the computer if you do not have a phone. Click Run>Run as Android Application. The emulator will take about 3 minutes to load. Your application should open immediately. If it does not, then go to the menu, and scroll down the icons to find your application. If you adjusted the code, then DO NOT close the emulator window… instead just click RUN again.

**Working with Activities**

**Writing Activity Class**

To create an activity, you must create a subclass of *Activity*. It is an extremely good idea to override these methods for the Activity class - onCreate(), onPause() – and less so for – onStop(), onResume(), onDestroy(). OnCreate is called when the activity is first initialized, onPause is called when the user leaves the activity but does not destroy the it (if you choose another activity on top of this one).

You must declare activities in the Manifest file, and declare the intent filters. If you do want the activities to be launched by other activities then you must define intent filters.

**Starting an Activity**

Intent intent = new Intent(this, <your activity class name>.class);

startActivity(intent);

You may also put in additional information into an intent using…Intent.putExtra(<string identifier>, value) …. There are many many versions of intent.putExtra() so check out the API. To receive the additional information from the called activity, then you can say Intent myIntent = getIntent(); and then myintent.getStringExtra(<string identifier>) assuming that the value is a String. You can use putExtra() multiple times on the same intent.

If you want to receive a result from the activity that you started, then use startActivityForResult() instead of startActivity(). Then in the called activity implement onAcitvityResult() callback method. This will return a result in an Intent to your onActivityResult() method, therefore onActivityResult() should have an Intent type in its parameter.

**Shutting down an Activity**

Use finish(). Or shut down a different activity using finishActivity()

**Working with Views**

A user interface for an activity is a hierarchy of views – objects derived from the *View* class. Each view is in control of a particular rectangle of space, and may be as small as a button.

“Widgets” are already-made views by Android (image, textfield, button) and “Layouts” are views from the ViewGroup that have a unique layout model (linearlayout, relative layout).

You can set the UI by passing in your layout (ViewGroup) into setContentView()

**Using the files**

**Src Folder**

Where your code and class files should be

**Bin Folder**

The most important file in this folder is the .apk file. This is generated when you run the Android emulator, and is the file that holds the actual application for the Android phone.

**Res Folder**  
Has drawable folder (holding images), layout folder (holding xml layouts of activity user interface) and values folder (xml files holding color or string values). Any file in these folders will be assigned a resource ID (a picture “picture.png” in the res/drawable directory will auto-generate a resource ID called R.drawable.picture of type int). Can save multiple XML files, for example, if you would like to have different string values in different languages. Upload all content here, especially images.

You should define animations, menus, styles, colors, and layout of activity user interfaces in XML files. You may have multiple XML files

**Gen Folder**

Has one file R.java. This is an auto-generated file that updates itself with all values declared in your XML files from your Res folder. This is where the “R” in R.drawable.blah comes in. Don’t modify this file. Often times when you have added content to the res folder, I will not show up in your application until the gen folder has updated.  
  
**Android Manifest File (AndroidManifest.xml)**  
Your application must declare all of its components in this file, or else the system will not know that these components exist. In this file you also must:

1. Declare all activities, services, and content providers. Broadcast receivers can be declared in manifest or created in code.
2. Identify user permissions the application needs such as Internet access or read-access to user’s contacts
3. Declare minimum API Level requirement for this application
4. Declare hardware and software features required for application (camera, bluetooth, multitouch screen, trackball)
5. Declare API libraries the application needs other than Android framework APIs
6. Declare any intent filters - the way the system identifies the correct components to respond to an intent.
7. Declare screen sizes your application supports

**Important stuff**

**Debugging**

A useful tool that allows you to debug similarly to that of System.out.println() is LogCat. To use you must:

1. Declare “import android.util.Log” in every class that you want to use this in
2. Declare private static String LOG\_TAG=”your class name or identifier” at the top of your class variables
3. When you would want use System.out.println(), instead use Log.v(LOG\_TAG, “your message here”)
4. When you run your program, you can open up LogCat by clicking this icon,. When the program runs, you should see your message pop up with the value you inputed for your LOG\_TAG