Android - Overview

- What is Android?
- Application Components
- Activity Life Cycles
- Homework Assignment #almost 1
  - Do the notepad tutorial (all 3 steps)
  - HW #1 assigned on Wednesday
What is Android?
Application Components

• Activities
  – Visual user interface
  – Hierarchy of Views

• Services
  – Background processes (playing music, etc.)

• Broadcast Receivers
  – Low battery, time zone change, etc.

• Content Providers
  – Allows data sharing between applications
Activating Components

• **ContentProvider**
  – Activated when targeted by a ContentResolver

• **Intents**
  – Start: Activities, Services, BroadcastReceivers
  – Activities, services: names the action and the data
  – BroadcastReceivers: names the action being announced.
AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?>
<manifest . . . >
  <application . . . >
    <activity
      android:name="com.example.project.FreneticActivity"
      android:icon="@drawable/small_pic.png"
      android:label="@string/freneticLabel" . . . >
    </activity> . . .
  </application>
</manifest>
Activities vs. Tasks

• Activity is a screen
• Task is a group of Activities
  – Not necessarily defined in the same Application.
  – Stack of activities. Activities can only be pushed and popped.
  – All activities in a task move as one, i.e. all go to background and or all to foreground at once.
Activity Lifecycle

1. **Activity starts**
   - **onCreate()**
     - User navigates back to the activity
     - **onStart()**
       - Activity is running
         - Another activity comes in front of the activity
         - **onPause()**
           - The activity comes to the foreground
           - The activity is no longer visible
           - **onStop()**
             - The activity comes to the foreground
             - **onDestroy()**
               - Activity is shut down
Activities Lifecycle

• Screen rotation will completely kill and restart your program.

\[ \text{onPause()} \rightarrow \text{onStop()} \rightarrow \text{onDestroy()} \]

A new instance of your application is created

\[ \text{onCreate()} \rightarrow \ldots \]
Activities – Saving State

- Primitives, parcelables, serialized objects
  - `onSaveInstanceState(Bundle outState)`
  - `onRestoreInstanceState()` or manually in `onCreate(Bundle savedInstanceState)`

- Objects
  - `onRetainNonConfigurationInstance()`
  - `getLastNonConfigurationInstance()`
Activities - Threads

• UI thread
  – Must be quick. Respond in less than 9 seconds.

• Background Threads
  – For long activities, downloading, etc..
  – Use AsyncTask
Views

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent">

<Button
android:id="@+id/add_button"
android:text="@string/add_file"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentBottom="true"
android:padding="15px"
android:textSize="8pt"
android:layout_weight="1"/>

<ListView
android:id="@android:id/list"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:layout_above="#id/upload_button"
android:layout_alignParentTop="true" />
</RelativeLayout>
Views

onCreate() {
    setContentView(R.layout.myLayout);
    // where myLayout is in {project}/res/layout/myLayout.xml

    Button b = (Button) findViewById(R.id.add_button);
    b.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            // do something interesting;
        }
    });
}
Activities – Dialogs

• Managed (Android)
  – onCreateDialog()
  – showDialog()
  – onPrepareDialog() - broken

• Self-Managed
  – Dialog m = new Dialog()
  – m.showDialog();
  – m.dismissDialog();
adb - your new best friend.

- **adb {-d or –e}**
  - devices – shows connected devices/emulators
  - shell – opens shell on device
  - push – push files to device
  - pull – pull files from device
  - logcat – display log output
- `Log.e(“tag”, “log entry”);`
Faking it on your local emulator

- `adb devices`
  - emulator-5554 (or similar)
- `telnet localhost 5554`
  - `geo fix 1 2`
  - `help`
Random tips

• Emulator not always seen by adb – restart
• Activity started by eclipse not same as Activity started from launch menu
• API Demos has examples of most of what you want to do
• Not everything works how you think it will (taking pictures). It’s a work in progress.
Threading Caveats

```java
int mResult;  Thread mThread;
Handler mHandler = new Handler();

onCreate() {
    mThread = new Thread() {
        public void run() {
            mResult = doSomethingExpensive();
            mHandler.post(results);
        }
    };
    mThread.start();
}
```

Orientation Change
Threading Caveats

```java
int mResult;  Thread mThread;
Handler mhandler = new Handler();

onCreate() {
    mThread = new Thread() {
        public void run() {
            mResult = doSomethingExpensive();
            mHandler.post(results);
        }
    }
    mThread.start();
}
```

Orientation Change

Activity²
Threading Caveats

```java
int mResult;  Thread mThread;

Handler mhandler = new Handler();

onCreate() {
    mThread = new Thread() {
        public void run() {
            mResult = doSomethingExpensive();
            mHandler.post(results);
        }
    }
    mThread.start();
}
```

Orientation Change

```
Activity₂

Thread t₂
```
Threading Caveats

onRetainNonConfigurationInstance() {
    return mThread;
}

onCreate() {
    ...
    mThread = (Thread) getLastNonConfigurationInstance();
    if (mThread == null) {
        mThread = new Thread();
        mThread.run();
    }
}

Activity

mResult

mHandler

Thread t

Orientation Change
Threading Caveats

onRetainNonConfigurationInstance() {
    return mThread;
}

onCreate() {
    ...
    mThread = (Thread) getLastNonConfigurationInstance();
    if (mThread == null) {
        mThread = new Thread();
        mThread.run();
    }
}

mResult
mHandler

Thread t₁

Activity₂
mResult₂
mHandler₂
public class MyActivity implements myListener {

    onResume() {
        mThread.setListener(this);
    }

    onDestroy() {
        mThread.setListener(null);
        t.start();
    }

    threadComplete() {
        // thread is done, display result
    }
}

Thread {
    myListener mListener;
    run () {
        // do stuff
        mListener.threadComplete();
    }
}