User Interface Design & Development

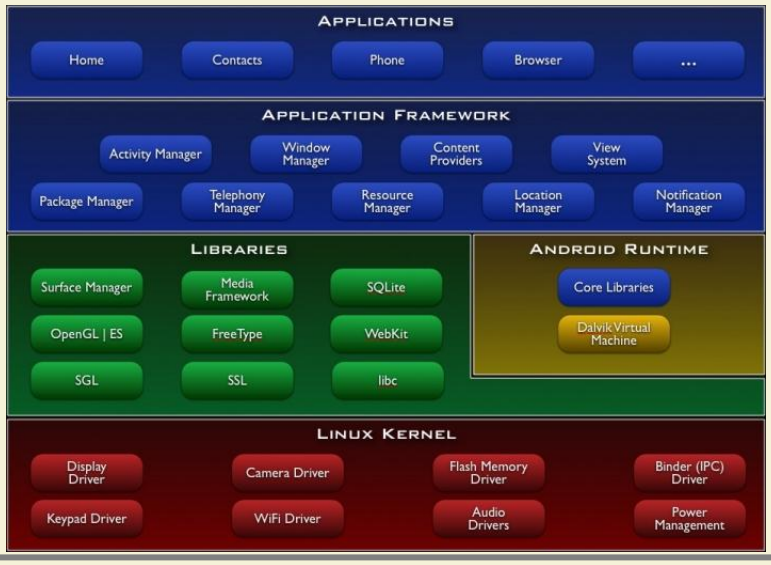
Lecture
Intro to Android

João Pedro Sousa
SWE 632, Fall 2011
George Mason University

features

- multitasking w/ just-in-time compiler for Dalvik-VM bytecode
- storage on SQLite
- networking
  - GSM/EDGE, Bluetooth, Wi-Fi, WiMAX, etc.
- messaging: SMS, MMS
- web browser: WebKit with Chrome engine
- media support: mp3, mp4, wav, jpeg, etc.
  - streaming support: RTP, html progressive, Adobe Flash, etc.
- sensors
  - multi-touch display, GPS, accelerometers, gyroscopes, cameras…

system layers



apps

- bundled as an Android package *file*.apk
  - may be downloaded to a phone or emulated in Eclipse
  - by default runs in its own process associated with a uId
    - for isolation, each process runs on its own VM
  - may share files with other apps with the same uId
  - may share a process and VM with other apps with the same uId
- app components may render services for other apps
  - no single entry point, i.e. main
  - no need to link other code: communication by *intents*

apps don't control their own process termination

- processes remain alive until OS terminates them
- decision based on
  - memory availability
  - importance
    - foreground: interacting with user
    - visible: on the screen but *paused*
    - service: executing, e.g. playing music, but not visible
    - background: not executing and not visible, i.e. *stopped*
    - empty: inactive app, just a cache to improve startup time
- upshot
  - user navigates back  instead of closing windows

types of components

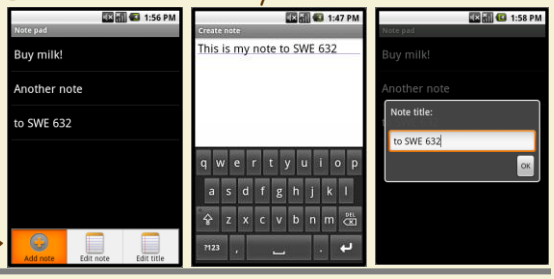
- *activity*, i.e. user interaction activity
  - a UI may include many
- *service*, i.e. ongoing background process
  - may expose a control api
- *broadcast receiver*,
  - reacts to *intents*, i.e. control msgs sent by activities & services
- *content provider*
  - facilitates data sharing among apps via SQLite

activity

- single focused thing i.e. interaction a user can do
- typically presented as full-screen window
  - may use *floating* window or be embedded in an *activity group*
  - examples from the NotePad sample code:
    - public class NotesList extends ListActivity
    - public class NoteEditor extends Activity
    - public class TitleEditor extends Activity

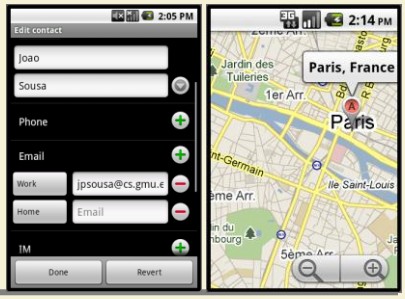
ListActivity ->

menu ->



activity
associated with layout resource

- public void setContentView (int layoutResID)
  - *inflates* the resource, i.e. shows it and its parents
- layouts may be created
  - in XML
  - programmatically
  - examples:

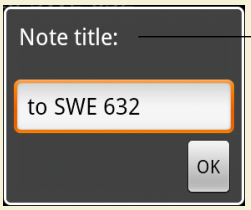


example
layout in XML NotePad.TitleEditor

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:orientation="vertical"
   android:paddingLeft="6dip"
   android:paddingRight="6dip"
   android:paddingBottom="3dip">

   <EditText android:id="@+id/title"
      android:maxLines="1"
      android:layout_marginTop="2dip"
      android:layout_width="wrap_content"
      android:ems="25"
      android:layout_height="wrap_content"
      android:autoText="true"
      android:capitalize="sentences"
      android:scrollHorizontally="true" />

   <Button android:id="@+id/ok"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="right"
      android:text="@string/button_ok" />
</LinearLayout>
```

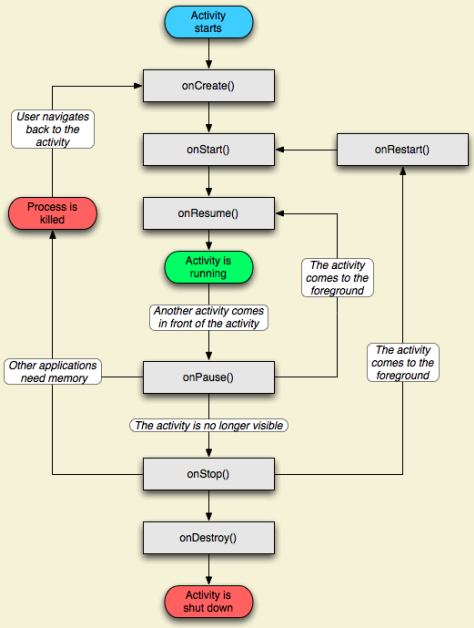



```
public void onCreate(…) {
…
  Button b = (Button) findViewById(R.id.ok);
  b.setOnClickListener(this); …
```

look at res.values.strings.xml

activity lifecycle

- activity may be
  - active/running: screen foreground
  - paused: lost focus but visible
  - stopped: not visible
- VM issues callbacks to an activity as the user interacts with phone
- upshot: an activity has little control over the flow of interaction
  - announces *intents* that will be picked up by other activities



intents

facilitate communication between activities

- example: the activity menu for NotesList generates intents…

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    …
    // This is our one standard application action -- inserting a new note into the list.
    menu.add(0, MENU_ITEM_INSERT, 0, R.string.menu_insert)
            .setShortcut('3', 'a')
            .setIcon(android.R.drawable.ic_menu_add); …
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   switch (item.getItemId()) {
      case MENU_ITEM_INSERT:
         // Launch activity to insert a new item
         startActivity(new Intent(Intent.ACTION_INSERT, getIntent().getData()));
         return true;
      } …
}
```

predefined String android.content.**Intent.ACTION_INSERT** = "android.intent.action.INSERT"

manifest file

declares app components and supported intents

- …which the NoteEditor catches…
  manisfest example: NotePad

```
<manifest …  package="com.example.android.notepad">
   <application android:icon="@drawable/app_notes"
                android:label="@string/app_name">
   <…>
      <activity android:name="NotesList" android:label="@string/title_notes_list">
         <…>
         <intent-filter>
            <action android:name="android.intent.action.VIEW" />
            <action android:name="android.intent.action.EDIT" /> …
         </intent-filter> …
      </activity>
      <activity android:name="NoteEditor" …>
         <intent-filter>
            <action android:name="android.intent.action.INSERT" /> …
   </application>
</manifest>
```

also the place to set up permissions to pass intents to other apps

intents
are processed onCreate()

- …and the NoteEditor processes

```
@Override
protected void onCreate(…) {
  …
  final Intent intent = getIntent();
  final String action = intent.getAction();

  if (Intent.ACTION_EDIT.equals(action)) {
        // Requested to edit: set that state, and the data being edited.
        …
     } else if (Intent.ACTION_INSERT.equals(action)) {
        // Requested to insert: set that state, and create a new entry in the container.
        …
}
```


getting started

- follow the instructions on the assignments page
  cs.gmu.edu/~jpsousa/classes/632/
  to download and start an emulation
  - note: the android emulator window has different looks for each API level
    - level 7 looks like this :

- then, get started on
  - Lab 1
    - based on NotePad