Development with Android

Computing Science and Mathematics
University of Stirling

Activities and Tasks

Activity Lifecycle

(from Android Developer site)

Activity Transitions

- Activities are notified of transitions via:
  - ***void*** *onCreate(Bundle savedInstanceState)*
  - ***void*** *onStart()*
  - ***void*** *onRestart()*
  - ***void*** *onResume()*
  - ***void*** *onPause()*
  - ***void*** *onStop()*
  - ***void*** *onDestroy()*

Activity Lifetimes

- Entire lifetime:
  - *onCreate* to *onDestroy*
  - e.g. while handling a persistent Internet connection
- Visible lifetime:
  - *onStart* to *onStop*
  - e.g. while handling a broadcast receiver
- Foreground lifetime:
  - *onResume* to *onPause*
  - frequent transitions between them, so must be lightweight

Activity Example - 1

```
package uk.ac.stir.cs.android;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class CountStates extends Activity {

 int paused = 0;
 int killed = 0;
 int stopped = 0;
 TextView textView = new TextView(this);

 private void setCounts() {
  textView.setText("paused: " + paused + " stopped: " +  stopped +
    " killed: "+ killed);
 }
```

(from San Diego State University site)

Activity Example - 2

```
public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 if (savedInstanceState != null) {
  paused = savedInstanceState.getInt("paused");
  killed = savedInstanceState.getInt("killed");
  stopped = savedInstanceState.getInt("stopped");
 }
 setCounts();
 setContentView(textView)
}
```


Activity Example - 3

```
protected void onStart() {
  super.onStart();
  setCounts();
}

protected void onStop() {
  stopped++;
  super.onStop();
}

protected void onResume() {
  super.onResume();
  setCounts();
}
```

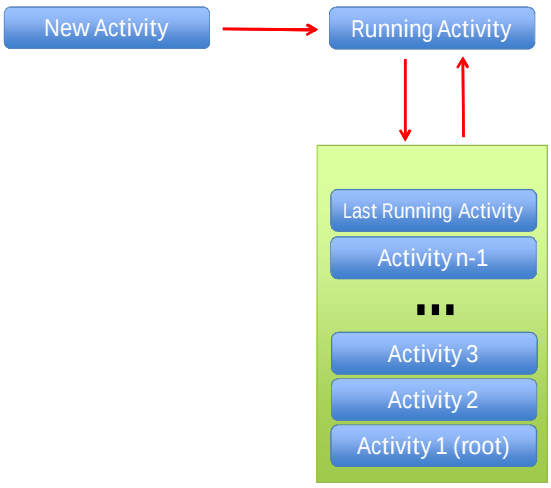
Activity Example - 4

```
protected void onPause() {
  paused++;
  super.onPause();
}

protected void onDestroy() {
  killed++;
  super.onDestroy();
}

protected void onSaveInstanceState(Bundle outState) {
  outState.putInt("paused", paused);
  outState.putInt("killed", killed);
  outState.putInt("stopped", stopped);
}
}
```


Activity Interplay

- One activity can start another:
  - including activities from other apps
- Consider a map application:
  - map activities exist, so no need to re-create them
  - an activity creates an intent and passes it to *startActivity*
  - the intent is caught by the map viewer's intent filter
  - the map viewer becomes the foreground activity
- As a result, going back from the map viewer returns to the previous activity

Activity Stack - 1



Return when back button pushed or running activity closed

May be removed to free resources

Activity Stack
(previous activities)

Activity Stack - 2

- Activities are never re-arranged:
  - can only be pushed and popped
- Switching to another activity multiple times?
  - multiple instances of that activity on the stack

Task that opens map viewer multiple times

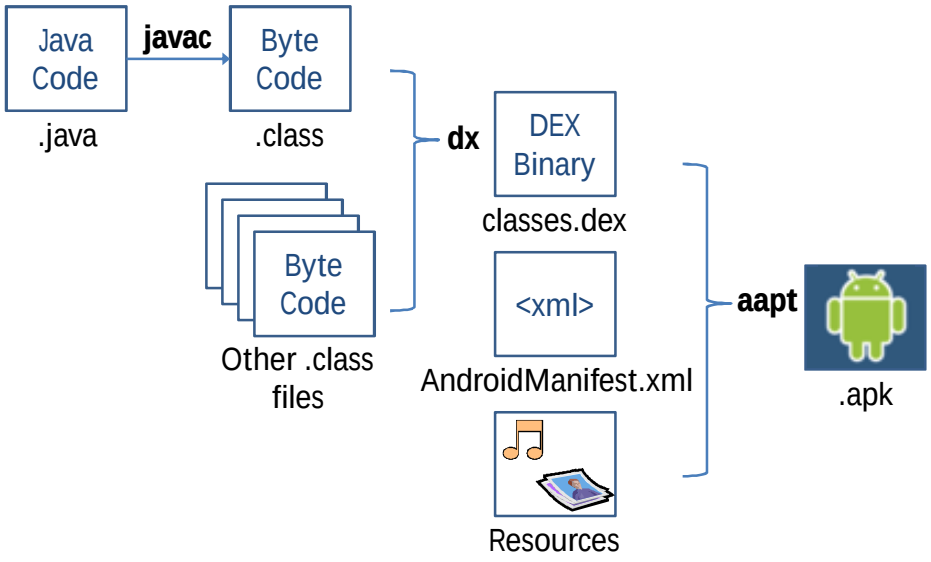
Launching Apps

- *AndroidManifest.xml*:
  - included with every app (APK is Android's JAR)
  - declares all components and resources
  - establishes window/process behaviour
  - sets permissions
  - lists required APIs
- At a minimum, the XML provides launch information
- The manifest is normally edited with an IDE (e.g. Android Studio)

User Interface Tools

- Tools ease user interface/XML definition, e.g.:
  - Android Studio
    http://developer.android.com/tools/studio/index.html
  - Droid Draw
    http://www.droiddraw.org

Putting it All Together



Developing for Mobile

- Low processing power
- Limited RAM
- Limited permanent storage
- Small screens
- High costs for data transfer
- Intermittent connectivity, slow data transfer rates, high latency, unreliable data connections
- Limited battery life

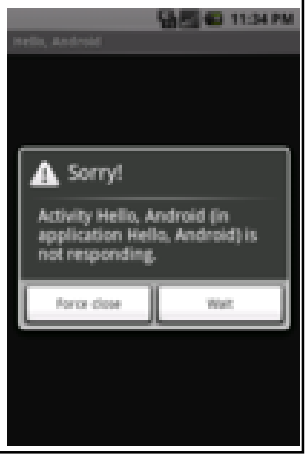
App priority and state

- Android apps do not have control over their own life cycles
- Android aggressively manages resources to ensure device responsiveness
- Android kills process/apps when needed

- Active Process – critical priority
- Visible Process – high priority
- Started Service Process
- Background Process – low priority
- Empty process

Approaches

- Be efficient
  - Optimise code so it runs quickly and responsively
  - Do not carry over assumptions from desktop development
  - Think about memory usage, object creation
- Be responsive
  - Android takes this very seriously (with Activity Manager and Window Manager)
  - Application must respond to any user action (key press, touch screen) within 5 sec
  - Broadcast receivers needs to return from onReceive within 10sec
  - Common problems: lengthy tasks in main thread, network & DB lookups, complex processing, file I/O
  - Use worker threads and Services



Approaches

- Ensure data freshness
  - Use multitasking in Android
  - Update data in background while app is not in use
  - Fresh application displays data user want to see quickly
  - Balance data update frequency with battery usage
- Provide accessibility
  - Not every user will be the same (as you)
  - Language
  - Screen size
  - Font size
  - Disabilities

Approaches

- Develop secure apps
  - Android apps have access to hardware and network
  - Distributed independently
  - use open source platform
  - Users need to take responsibility with which apps they install and what permissions they allow
  - Android model provides sandboxes to every app, restricts access to services etc by requiring apps to declare permissions they require. Users are shown these requirements during installation
  - Minimize data your app uses and permissions it requires
  - Require permission for every service you publish or Intents you broadcast, do not leak secure information to other apps (location data)
  - Take care when accepting input from external sources (Internet, BT, SMS)

Approaches

- Seamless user experience
  - Consistent user experience
  - Start, stop, transition instantly
  - Speed/responsiveness of app should not degrade with duration it's run
  - Application should present a consistent user interface regardless of being restarted or resumed
  - Applications should interact seamlessly using intents etc
  - Applications should be intuitively to use
  - Persist data between sessions
  - Suspend tasks which use processing cycles, network bandwidth, battery life when app is not visible
  - Use services for things which need to continue running in the background
  - When app is brought back to the front it should return to its last visible state