CMSC498G
September 22, 2010
Caleb Strockbine

Activity
What is an Activity?
What is its purpose in the Android ecosystem?
How is the class defined?
What methods does it support?
The Activity Class

- Activities encapsulate “a single, focused thing that the user can do.”
- They present a visual interface to the user
  - Have a default window
  - Window content implemented in subordinate Views
Class Hierarchy

- java.lang.Object
  - android.content.Context
    - android.content.ContextWrapper
      - android.view.ContextThemeWrapper
        - android.app.Activity
Object

- Java root class
- Example methods
  - `toString()`
  - `hash()`, `equals()`
  - `wait()`, `notify()`
Context

- Interface to application environment
- Example methods
  - `fileList()` : Get list of files in the app package
  - `startService()` Start/stop services
  - `openOrCreateDatabase()` : create private database
  - Access resources and system services
ContextWrapper

- Proxy for Context
  - For example, can substitute Context when testing
- Example method:
  - attachBaseContext()
ContextThemeWrapper

- Allows application theme to be modified
- Example method
  - `setTheme()`
Task

- A task is a stack of activities
  - not necessarily stored in the same application package
- Android manages the task stack
  - Launching activity places it on top of the stack
  - Hitting back button pops activity off the stack
Activity States

- active - visible, not ready to run
- running - visible, has focus
- paused - visible, no focus, can terminate
- stopped - not visible, can terminate
- `onCreate()`
- `onStart()`
- `onResume()`
- `onPause()`
- `onStop()`
- `onDestroy()`
- Discuss each lifecycle method.
- When is each called?
- What do you know at that point?
onCreate()

- Call the inherited onCreate()
- Find resources by their IDs
- Set the activity’s content view
- Configure views as necessary
- Restore the UI to its previous state
onCreate()
onResume()

- Called just before the activity becomes visible
- If your activity uses a lot of resources when running, it’s better to get them here and release them in onPause() so that they’re not tying up memory when not used
onPause()

- Clean up any resources the activity won’t need while it’s in the background
- Save the current state in case the activity resumes
- Save or finalize any data in case the activity doesn’t resume
Activities are launched with intents
   - say Intents come later

2 methods
   - startActivity
   - startActivityForResult
Using an Activity

- Create an Intent that specifies the activity itself or a generic intent name
- Optional: Add extra info to the intent
- Call startActivity(), passing in the intent
- Or, if you want control back when the activity completes, call startActivityForResult()
- finish() is mostly for startActivityForResult()
Ending an Activity

- When the activity is done, call `finish()`
- `isFinishing()` returns false if the activity is being killed by the system, true if it’s ending due to `finish()` being called
“Configuration” encompasses all sorts of things: input device, orientation, etc.

- If the device’s configuration changes, the front activity is killed and started over.
- The user shouldn’t be aware that this is happening, so it’s particularly important to preserve the state of the activity.
- Give example of saving state
- See Android sample code for example
comment

- For exercise, use something more helloworld like -
- You want a simple, focused lab
Splash screen

- Create an activity that simply displays an image for a few seconds and then starts another activity
- The activity itself is very simple, but this is a good exercise to get some experience with AndroidManifest.xml, Eclipse, etc.
- Suggest using the Tic Tac Toe sample project to get started.
Splash Screen seems like a bad example. Wouldn’t you just display a view for a while and then kill it and display you’re main window?
Preserve State

- Modify the Tic Tac Toe project such that it saves and restores the state of a game
- Modify the MainActivity class such that it passes a game state in the intent when it starts a GameActivity
- Make GameActivity return its state so that the MainActivity can save it