Programming
Handheld Systems

Adam Porter
The Activity Class
Today’s topics

The activity class
The task backstack
The activity lifecycle
Starting activities
Handling configuration changes
Activity

Provides a visual interface for user interaction
Activity

Each activity typically supports one focused thing a user can do, such as:

- Viewing an email message
- Showing a login screen
Activity

Applications often comprise several activities
Navigation through Activities

Android supports navigation in several ways:

Tasks

The task backstack

Suspending & resuming activities
Tasks

A Task is a set of related Activities
These related activities don’t have to be part of the same application
Most tasks start at the home screen

Task Backstack

When an Activity is launched, it goes on top of the backstack
When the Activity is destroyed, it is popped off the backstack
Task Backstack

Activity 1

Start Activity 2

Activity 2

Start Activity 3

Activity 3

Start Activity 4

Back Stack

Activity 1

Activity 2

Activity 3

Activity 1

Activity 2

Activity 1

BACK key
The Activity lifecycle

Activities are created, suspended, resumed & destroyed as necessary when an application executes
The Activity lifecycle

Some of these actions depend on user behavior

Some depend on Android

  e.g., Android can kill activities when it needs their resources
Activity lifecycle States

**Resumed/Running** – visible, user interacting

**Paused** – visible, user not interacting, can be terminated*

**Stopped** – not visible, can be terminated
The Activity Lifecycle methods

Android announces activity lifecycle state changes to activity by calling specific activity methods.
Some Activity Callback Methods

protected void onCreate (Bundle savedInstanceState)
protected void onStart()
protected void onResume()
protected void onPause()
protected void onRestart()
protected void onStop()
protected void onDestroy()
The Activity Lifecycle

- Activity Launched
  - onCreate() → onStart() → onResume()
  - onPause() → onStop() → onDestroy()
  - onRestart()

- Activity Running
The Activity Lifecycle

Activity Launched

onCreate()
onStart()
onResume()

onPause()
onStop()
onDestroy()

Activity Running

onRestart()

App. Proc. Killed

Activity Shut Down
The Activity Lifecycle

- Activity Launched
- onCreate()
- onStart()
- onResume()
- onPause()
- onStop()
- onDestroy()

Entire Lifetime
The Activity Lifecycle

Activity Launched
- onCreate()
  - onStart()
    - onResume()
      - onPause()
        - onStop()
          - onDestroy()
            - Activity Shut Down

Visible
- onRestart()
The Activity Lifecycle
The Activity Lifecycle

Activity Launched

onCreate()
onStart()
onResume()
Activity Running

onPause()
onStop()

onDestroy()

App. Proc. Killed

onRestart()

Activity Shut Down
The Activity Lifecycle
The Activity Lifecycle

- Activity Launched
  - onCreate()
  - onStart()
  - onResume()
  - Activity Running
  - onPause()
  - onStop()
  - onDestroy()
  - Activity Shut Down

  - onRestart()
onCreate()

Called when Activity is created
Sets up Initial state

Call super.onCreate()
Set the Activity’s content view
Retain reference to UI views as necessary
Configure views as necessary
onRestart()

Called if the Activity has been stopped and is about to be started again

Typical actions

Special processing needed only after having been stopped
onStart()

Activity is about to become visible

Typical actions

Start when visible—only behaviors

Loading persistent application state
onResume()

Activity is visible and about to start interacting with user

Typical actions

Start foreground-only behaviors
onPause()

Focus about to switch to another Activity

Typical actions
  Shutdown foreground-only behaviors
  Save persistent state
onStop()

Activity is no longer visible to user
may be restarted later

Typical actions
Cache state

Note: may not be called if Android kills your application
onDestroy()

Activity is about to be destroyed

Typical actions

Release Activity resources

Note: may not be called if Android kills your application
@Override
protected void onStart() {
    super.onStart();
    Log.i(TAG, "The activity is visible and about to be started.");
}

@Override
protected void onRestart() {
    super.onRestart();
    Log.i(TAG, "The activity is visible and about to be restarted.");
}

@Override
protected void onResume() {
    super.onResume();
    Log.i(TAG, "The activity is and has focus (it is now \"resumed\")");
}
Starting Activities

Create an Intent object specifying the Activity to start
Starting Activities

Pass newly created Intent to methods, such as:

startActivity()

startActivityForResult()

Invokes a Callback method when the called Activity finishes to return a result
setContentView(R.layout.main);

// Initialize UI elements
final EditText addrText = (EditText) findViewById(R.id.location);
final Button button = (Button) findViewById(R.id.mapButton);

// Link UI elements to actions in code
button.setOnClickListener(new Button.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            String address = addrText.getText().toString();
            address = address.replace(' ', '+');
            Intent geoIntent = new Intent(
                android.content.Intent.ACTION_VIEW, Uri.parse("geo:0,0?q=" + address));
            startActivity(geoIntent);
        } catch (Exception e) {
        }
    }
});
MapLocationFromContacts

Similar to MapLocation, but gets address from contacts database
Activity setResult()

**Started Activity can set its result by calling Activity setResult()**

- public final void setResult (int resultCode)
- public final void setResult (int resultCode, Intent data)
Activity.setResult()

resultCode (an int)

RESULT_CANCELED
RESULT_OK
RESULT_FIRST_USER

Custom resultCodes can be added
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    final Button button = (Button) findViewById(R.id.mapButton);
    button.setOnClickListener(new Button.OnClickListener() {
        @Override
        public void onClick(View v) {
            try {
                Intent intent = new Intent(Intent.ACTION_PICK,
                                          ContactsContract.Contacts.CONTENT_URI);
                startActivityForResult(intent, PICK_CONTACT_REQUEST);
            } catch (Exception e) {
            }
        }
    });
}
Configuration Changes

Device configuration can change at runtime

Keyboard, orientation, locale, etc.

On configuration changes, Android usually kills the current Activity & then restarts it
Configuration Changes

Activity restarting should be fast

If necessary you can:

- Retain an Object containing important state information during a configuration change
- Manually handle the configuration change
Retaining an Object

Hard to recompute data can be cached to speed up handling of configuration changes.

Override `onRetainNonConfigurationInstance()` to build & return configuration object.

Will be called between `onStop()` and `onDestroy()`
Retaining an Object

Call `getLastNonConfigurationInstance()` during `onCreate()` to recover retained object.

**Note:** These methods have been deprecated in favor of methods in the `Fragment` class (discussed in later classes).
Manual Reconfiguration

Can prevent system from restarting Activity
Declare the configuration changes your Activity handles in AndroidManifest.xml file, e.g.,

<activity android:name=".MyActivity"
    android:configChanges="orientation|screensize|keyboardHidden"...>
Manual Reconfiguration

When configuration changes, Activity’s onConfigurationChanged() method is called
Passed a Configuration object specifying the new device configuration
Next Time

The Intent Class