Activities

https://developer.android.com/guide/components/activities.htm

Repo: https://github.com/karlmorris/AndroidBasicActivities
Overview

- What is an Activity
- Starting and stopping activities
- Introduction to Intents and context
- Tasks and the Back Stack
- The activity lifecycle
Activities

- Provides a mode of interaction for application users
- Extends the Activity class or another subclass
- A single activity should be used for a single action/operation
- Applications generally consist of one or more activities
- One activity is delegated as the Main
  - The first activity launched when the app is started
Activities

- Declared in the application manifest

```xml
<manifest ... >
  <application ... >
    <activity android:name=".ExampleActivity" />
    ...
  </application ... >
  ...
</manifest>
```

- `android:name` is required
- Many attributes supported such as `label`, `theme`, etc.
Starting an Activity

• An activity can be started by calling
  – startActivity(intent)
  – startActivityForResult(intent, requestCode)
• **intent** specifies the activity to start, or the action required
• **requestCode** allows us to determine which request has returned
  – Managed in onActivityResult(…)
• An activity can start activities from other applications
Context

- Context represents (provides a handle to) the application environment
- Can be obtained from several sources, but capabilities vary
Context

• Uses for context include
  – Loading activities and other components
  – Creating views
  – Accessing application resources
  – Accessing a shared/system service
Context

- You can get context from
  - an activity
  - a service
  - the application
  - Android
Intents

• A messaging object used to request an action
• Generally used to
  – Start an activity
  – Start a service
  – Send a broadcast
• Two types of intents
  – Explicit - specify a component
  – Implicit - do not specify a component
Intents in Action

An intent passed between two activities
Closing an Activity

- An activity can close itself using `finish()`
- An activity can close another activity that is started using `finishActivity(requestCode)`
- Android may also close activities in the background to reclaim memory
Activity Lifecycle

• The states of an activity between creation and destruction (inclusive) is its lifecycle

• Affected by
  – Other activities
  – Task
  – Back Stack

• Understanding the lifecycle is imperative for building flexible, responsive applications
Why understand the Activity Lifecycle

- Ensure app doesn't crash when user receives a phone call or switches to another app
- Prevent consumption of resources when user is not using app
- Persist user progress if they leave app to return later
- Does not crash or lose progress when device orientation changes
**Activity lifecycle**

1. **Activity launched**
   - onCreate()
   - onStart()
   - onResume()

2. **Activity running**
   - User navigates to the activity
   - Another activity comes into the foreground
   - Apps with higher priority need memory
   - onPause()
   - The activity is no longer visible
   - onStop()
   - The activity is finishing or being destroyed by the system
   - onDestroy()
   - Activity shut down

3. **App process killed**
   - User returns to the activity
   - User navigates to the activity
Activity Lifecycle

- **Created**
  - onCreate()
  - onStart()

- **Started** (visible)
  - onResume()
  - onStart()
  - onRestart()

- **Resumed** (visible)
  - onResume()

- **Paused** (partially visible)
  - onPause()
  - onStop()

- **Stopped** (hidden)
  - onRestart()
  - onDestroy()

- **Destroyed**
Static and Transient states

- **Static**
  - Resumed
    - When your activity is running
  - Paused
    - Activity is partially obscured
  - Stopped
    - Activity is in the background

- **Transient**
  - Created
  - Started
Implementing Lifecycle callbacks

public class ExampleActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // The activity is being created.
    }

    @Override
    public void onStart() {
        super.onStart();
        // The activity is about to become visible.
    }

    @Override
    public void onResume() {
        super.onResume();
        // The activity has become visible (it is now "resumed").
    }

    @Override
    protected void onPause() {
        super.onPause();
        // Another activity is taking focus (this activity is about to be "paused")
    }

    @Override
    public void onStop() {
        super.onStop();
        // The activity is no longer visible (it is now "stopped")
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        // The activity is about to be destroyed.
    }
}
Activities cont.

• Launch new activities using intents
• When launching, an activity A can pass data to another activity B
• When closing, and activity B can return a status and data to its parent activity
Back Stack

- Activities are managed using a structure called the Back Stack
Tasks

- A task is a collection of activities that work together to perform a job
- The device home screen is the starting point for most tasks
- When a user clicks an app icon, a task is created and the Main activity is launched
  - If the task already exists it's brought to the foreground
- Each task has its own Back Stack
Activity Launch Modes

• standard
  – Default launch mode – there can be multiple instances of the activity in a task, and instances in multiple tasks

• singleTop
  – Like standard, except a new instance will not be launched if an instance is already on top of the stack

• singleTask
  – Only one instance of the activity can exist. A new task is started when created, or brought to the foreground if one already exists

• singleInstance
  – Similar to singleTask, except the instance is always opened in a new task with no other activities
Example of singleTask