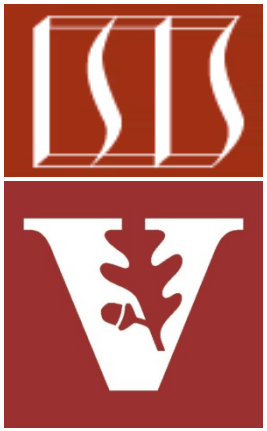
Overview of Activities



Douglas C. Schmidt
d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

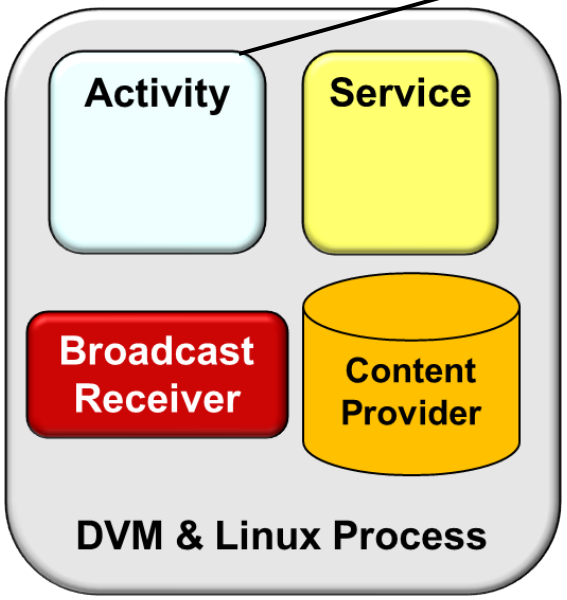
Institute for Software
Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA

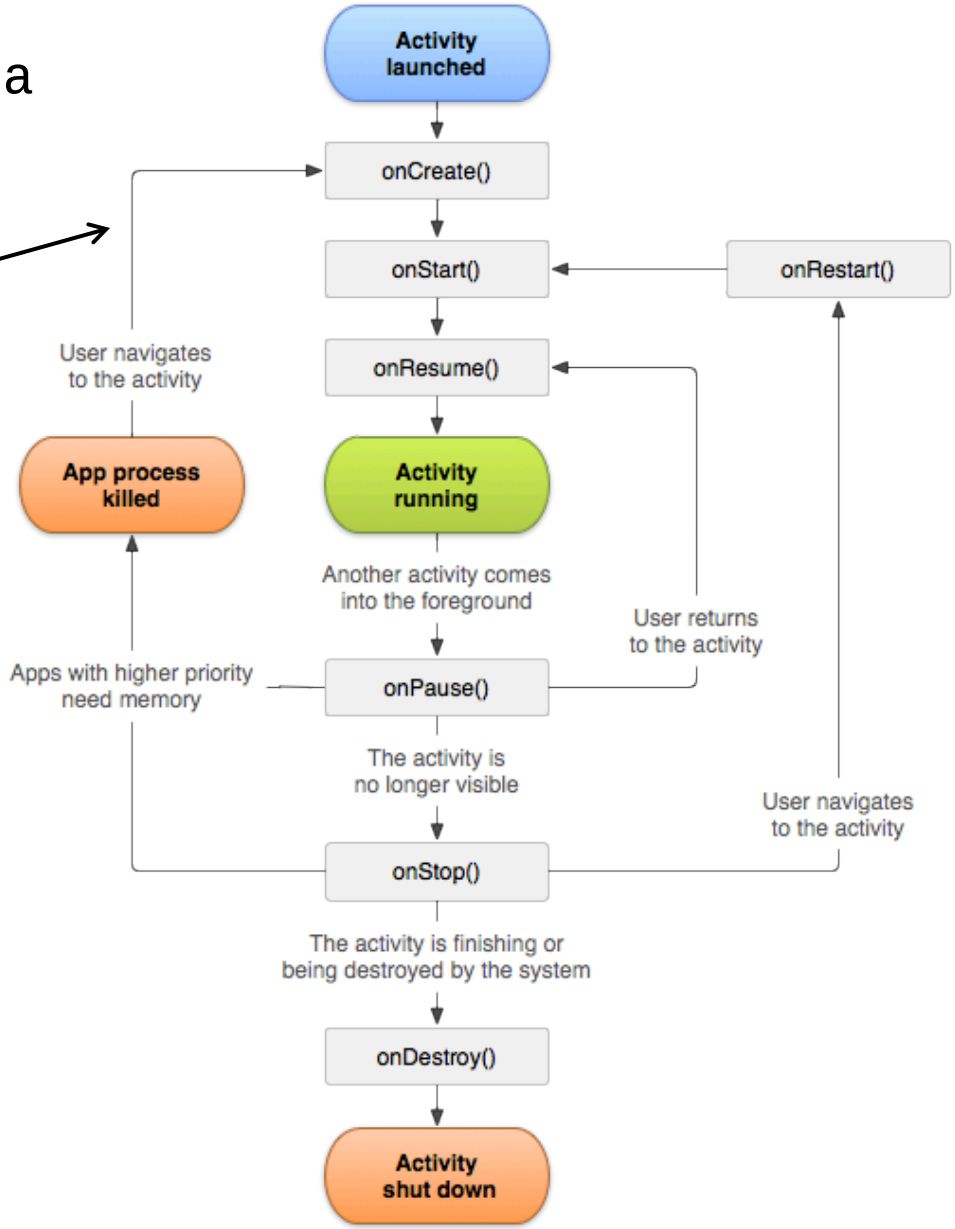
CS 282 Principles of Operating Systems II

Systems Programming for Android

Learning Objectives of this Module

- Understand how an Activity provides a visual interface for user interaction





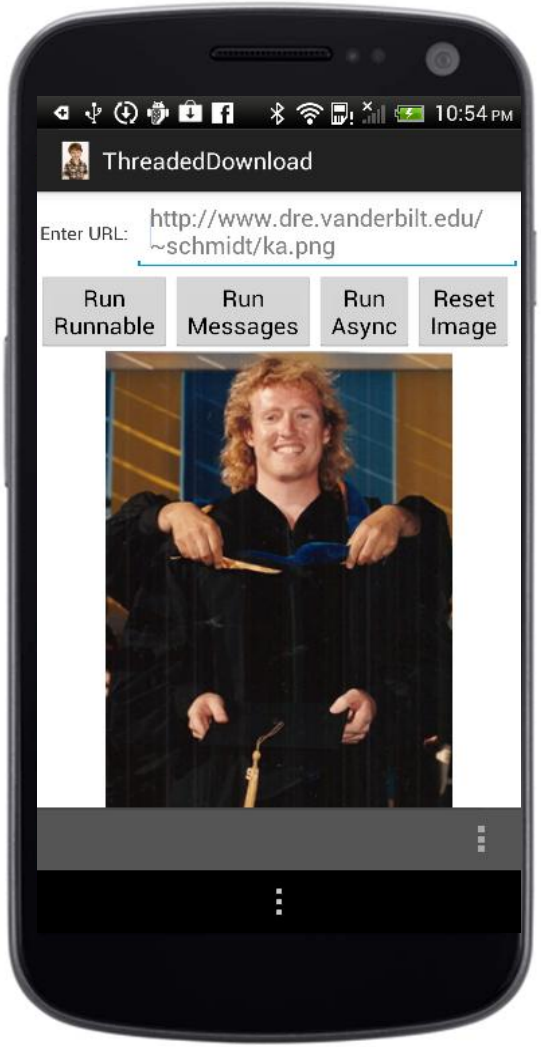
We'll emphasize commonalities & variabilities in our discussion

Overview of an Activity

- An Activity provides a visual interface for user interaction

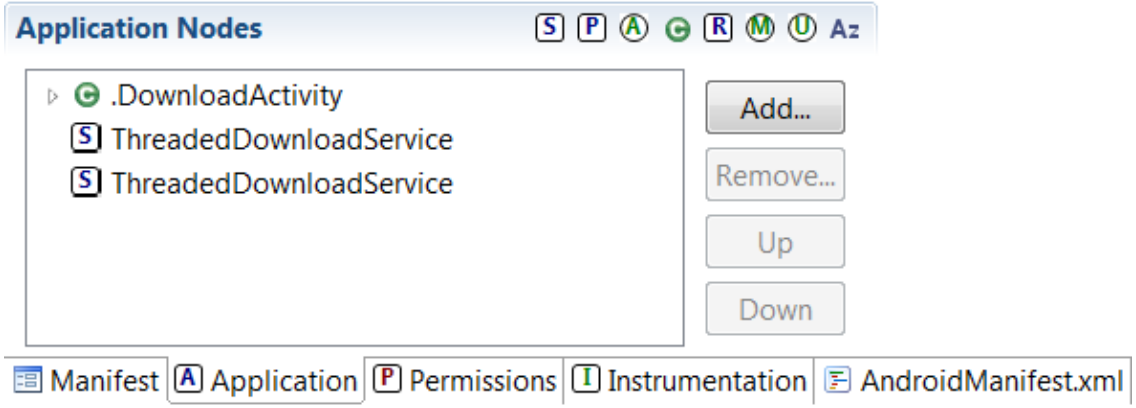
Overview of an Activity

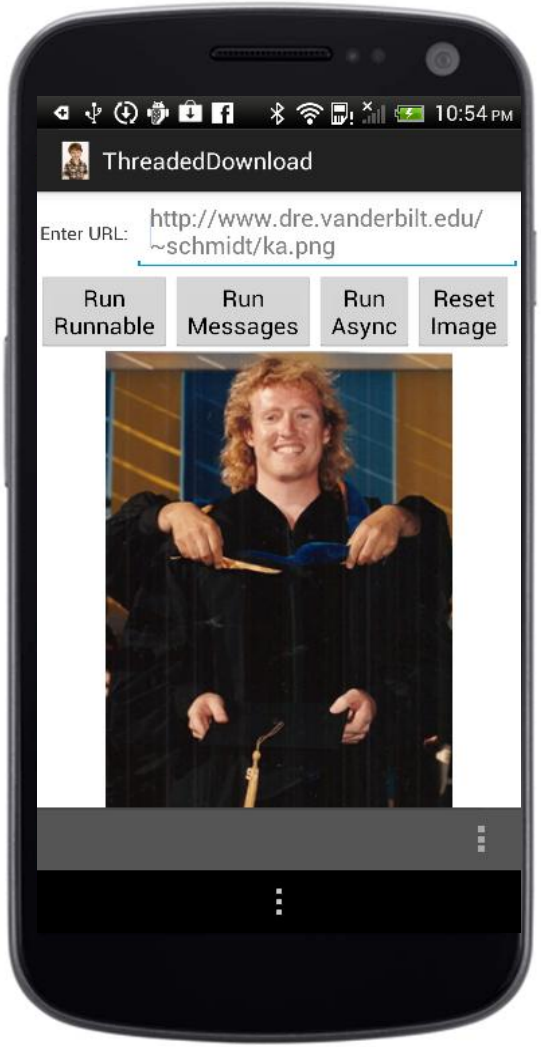
- An Activity provides a visual interface for user interaction
- Typically supports one thing a user can do, e.g.:
  - Show a login screen
  - Read an email message
  - Compose a text message
  - View a contact
  - Browse the Internet
  - etc.



Overview of an Activity

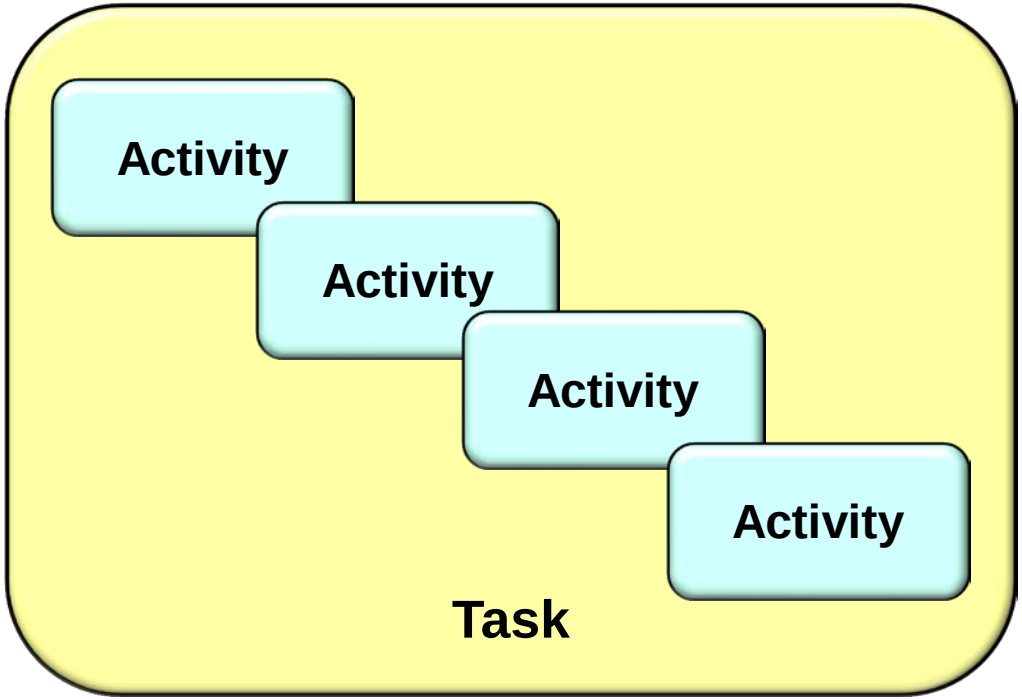
- An Activity provides a visual interface for user interaction
- Typically supports one thing a user can do, e.g.:
  - Show a login screen
  - Read an email message
  - Compose a text message
  - View a contact
  - Browse the Internet
  - etc.
- Applications can include one or more activities





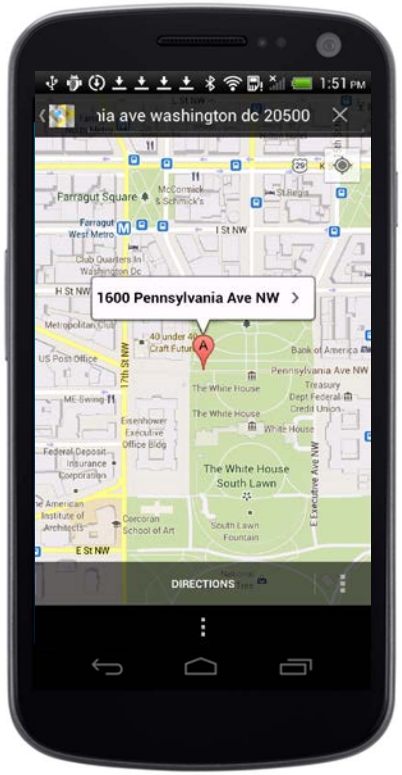
Tasks

- A Task is a chain of related Activities
  - Task are not necessarily provided by a single app



Tasks

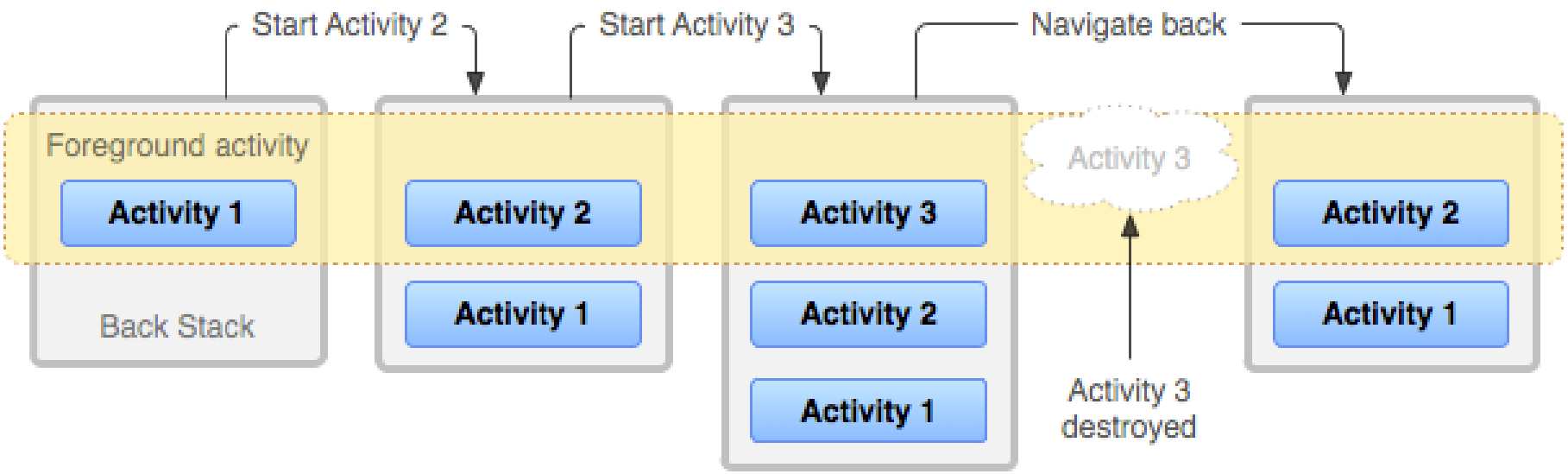
- A Task is a chain of related Activities
  - Task are not necessarily provided by a single app
- Tasks give the illusion that multiple (often unrelated) Activities were developed as part of the same app



developer.android.com/guide/topics/fundamentals/tasks-and-back-stack.html

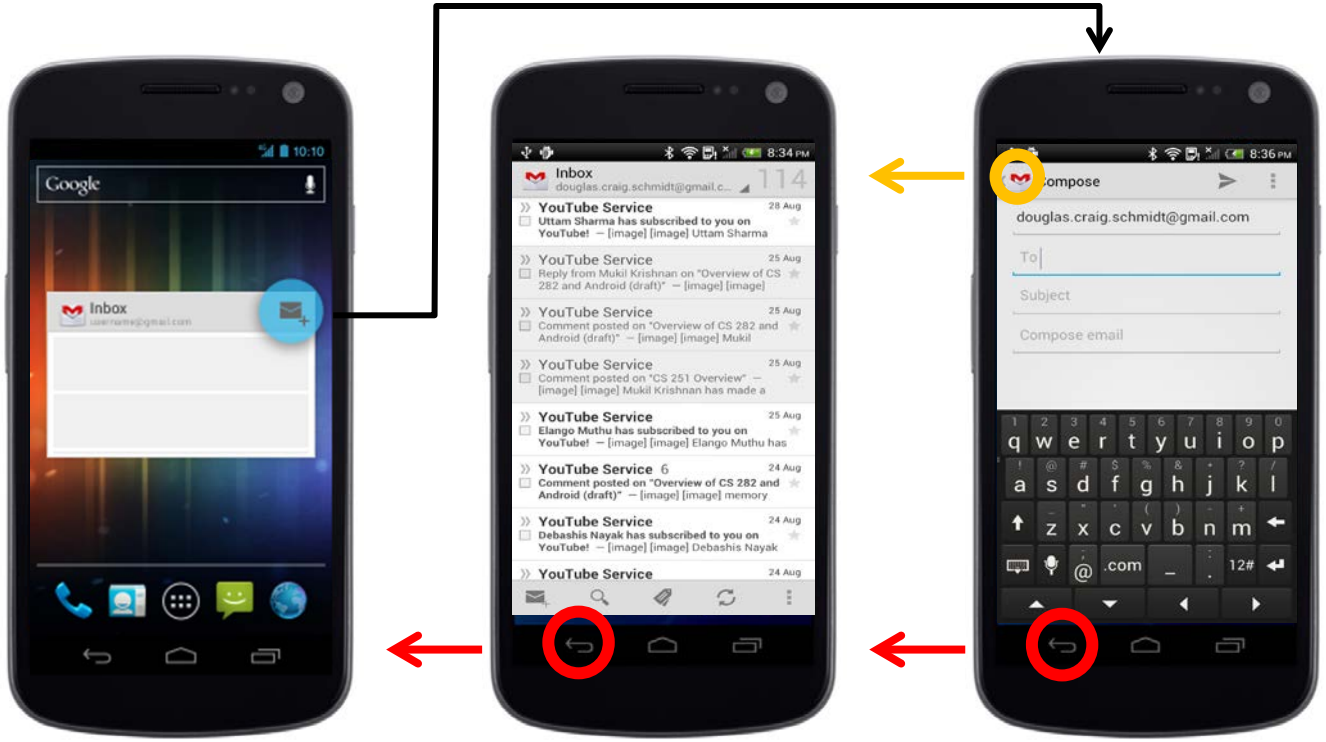
Tasks

- The task's Activity objects are stored on a "back stack" with the currently running Activity at the top



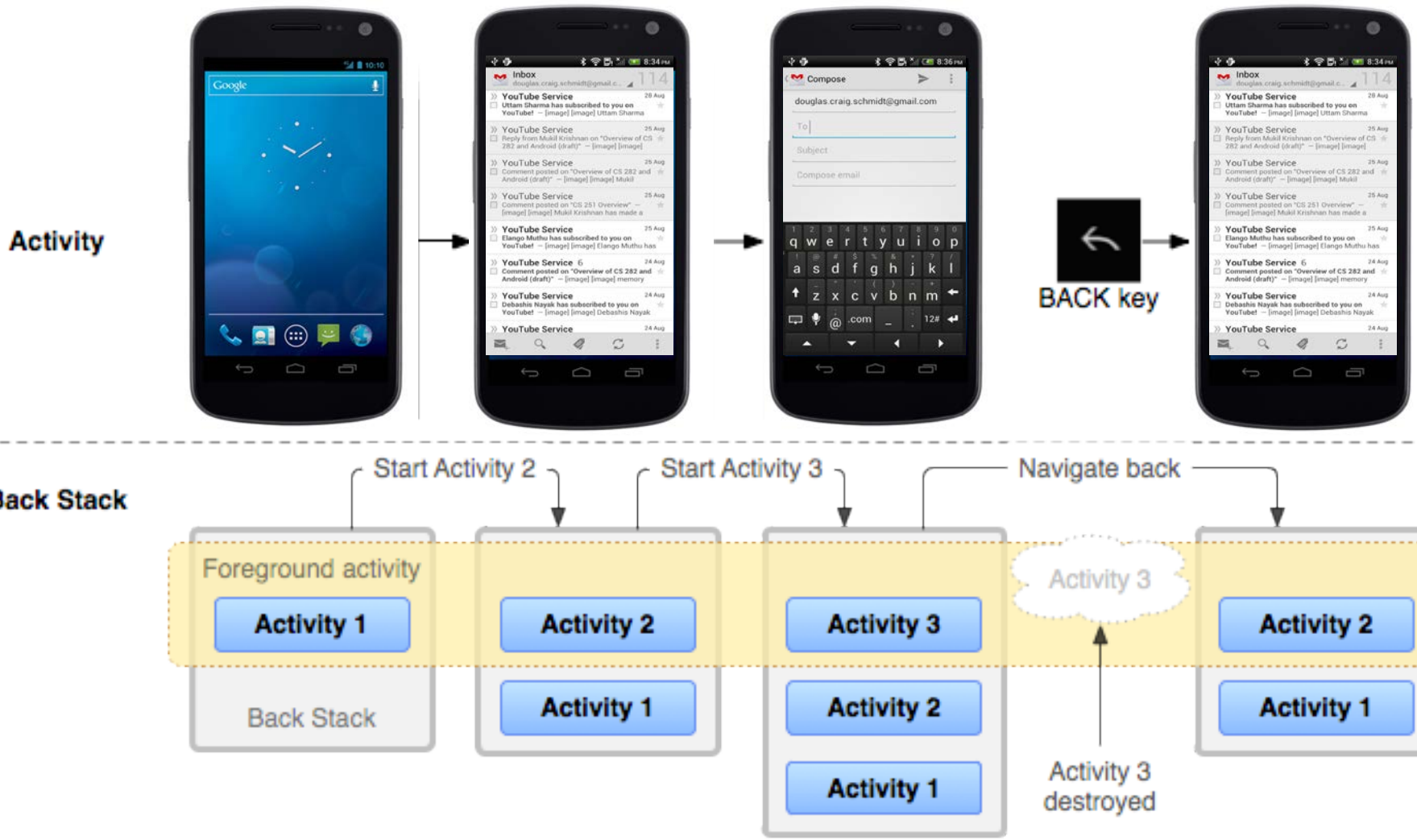
Tasks

- The task's Activity objects are stored on a "back stack" with the currently running Activity at the top
- At runtime
  - Launching an Activity places it on top of the stack
  - Hitting the BACK button pops current activity off the stack



developer.android.com/guide/topics/fundamentals/tasks-and-back-stack.html

Task Stack



developer.android.com/guide/topics/fundamentals/tasks-and-back-stack.html

Implementing an Activity

- Implementing an Activity involves several steps, e.g.:
  - Inherit from Activity class

```
public class MapLocation
             extends Activity {

   ...
}
```

Implementing an Activity

- Implementing an Activity involves several steps, e.g.:
  - Inherit from Activity class
  - Override selected lifecycle hook methods

```
public class MapLocation
             extends Activity {
   protected void onCreate
     (Bundle savedInstanceState);
   protected void onStart();
   protected void onRestart();
   protected void onResume();
   protected void onPause();
   protected void onStop();
   protected void onDestroy();
   ...
}
```

Implementing an Activity

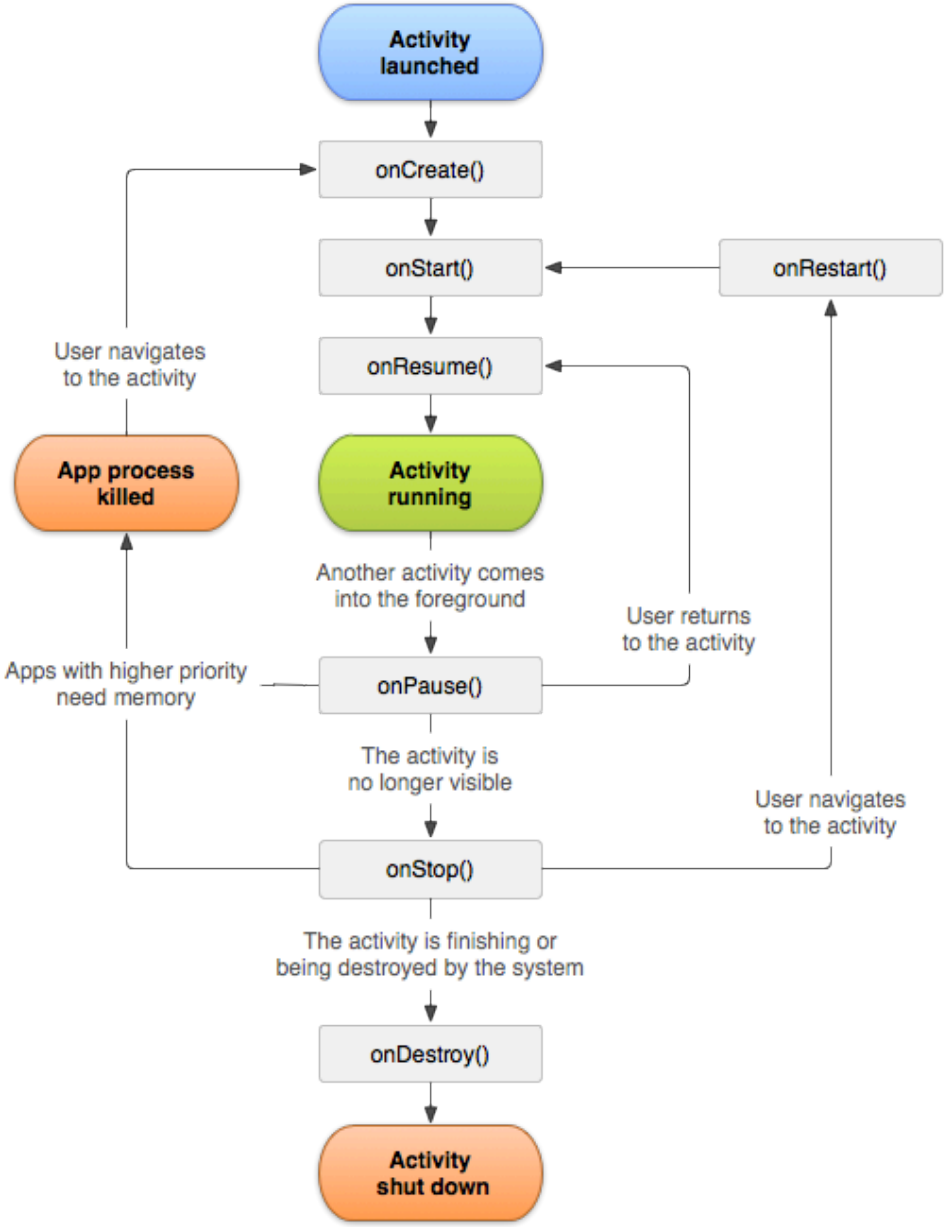
- Implementing an Activity involves several steps, e.g.:
  - Inherit from Activity class
  - Override selected lifecycle hook methods
  - Include Activity in the config file AndroidManifest.xml
  - etc.

```
public class MapLocation
             extends Activity {
   protected void onCreate
     (Bundle savedInstanceState);
   protected void onStart();
   protected void onRestart();
   protected void onResume();
   protected void onPause();
   protected void onStop();
   protected void onDestroy();
   ...
}
```

```
<activity
 android:name="course.examples.Activity.SimpleMapExample.MapLocation"
 android:label="Map A Location">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name=
                                "android.intent.category.LAUNCHER" />
            </intent-filter>
</activity>
```

Implementing an Activity

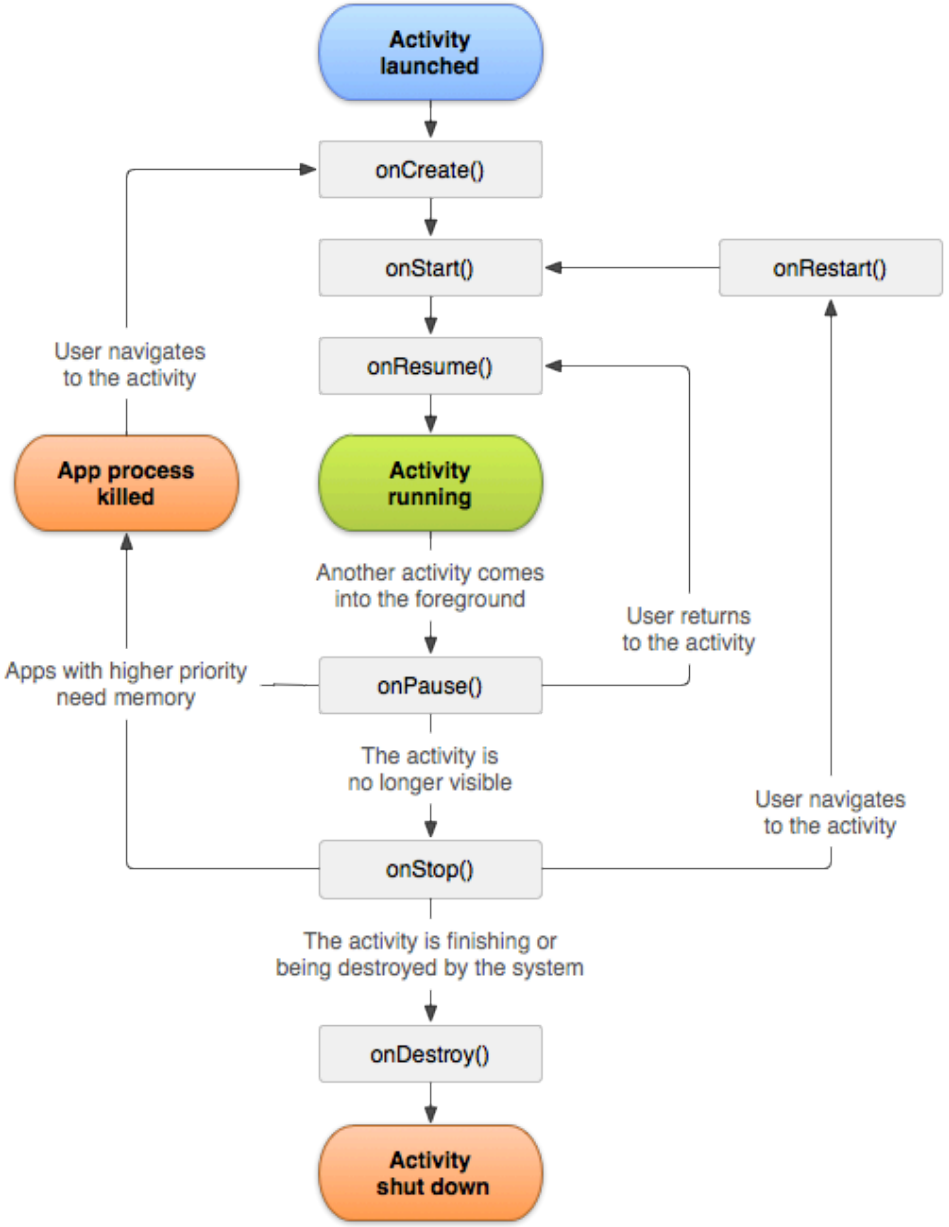
- Implementing an Activity involves several steps
- Android communicates state changes to an Activity by calling its lifecycle hook methods



Implementing an Activity

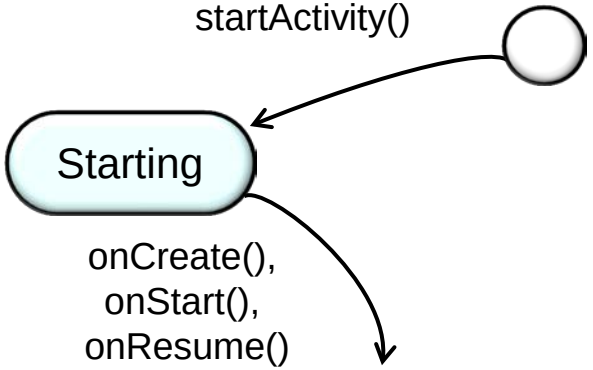
- Implementing an Activity involves several steps
- Android communicates state changes to an Activity by calling its lifecycle hook methods

- **Commonality**: Provides common interface for interacting with user, including operations performed when moving between lifecycle states
- **Variability**: Subclasses can override lifecycle hook methods to do necessary work when an Activity changes state



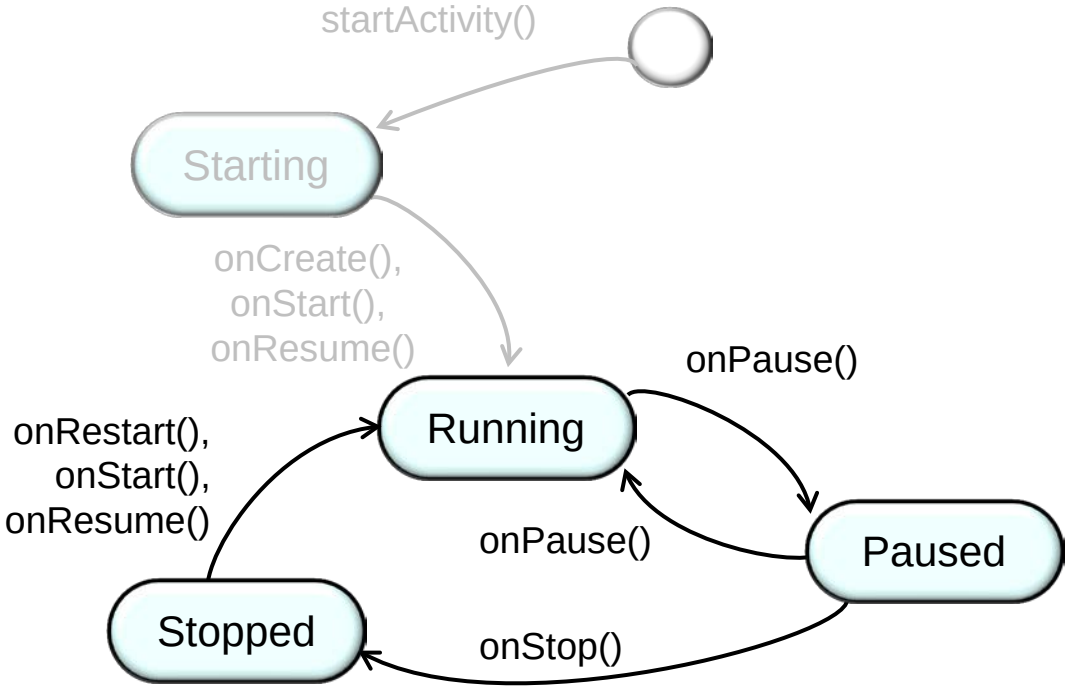
Activity Lifecycle States

- **Activity starting** – Initialization steps



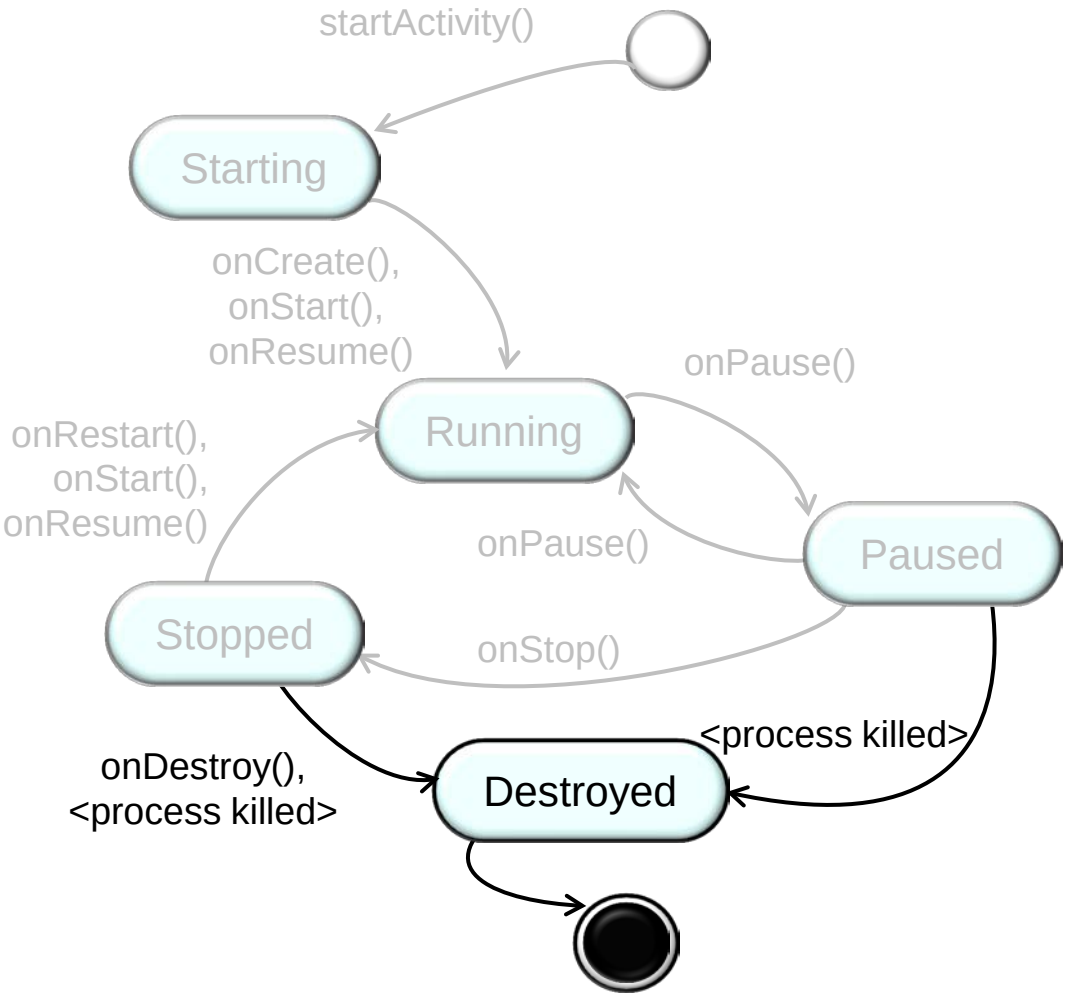
Activity Lifecycle States

- **Activity starting** – Initialization steps
- **Activity running**
  - *Running* – visible, has focus
  - *Paused* – visible, does not have focus, can be terminated
  - *Stopped* – not visible, does not have focus, can be terminated



Activity Lifecycle States

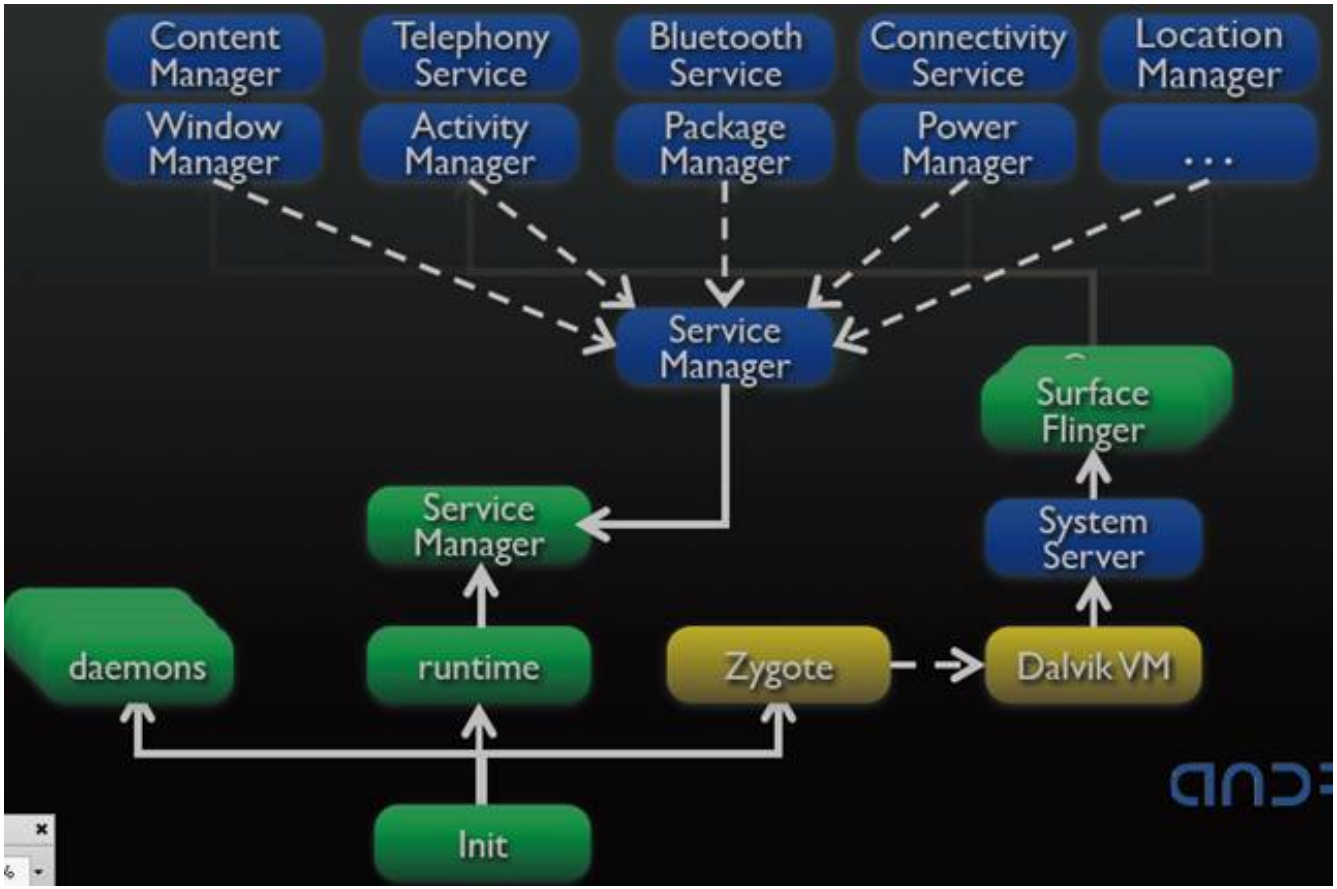
- **Activity starting** – Initialization steps
- **Activity running**
  - *Running* – visible, has focus
  - *Paused* – visible, does not have focus, can be terminated
  - *Stopped* – not visible, does not have focus, can be terminated
- **Activity shut down** – Voluntarily finished or involuntarily killed by the system



See developer.android.com/guide/components/activities.html for more info

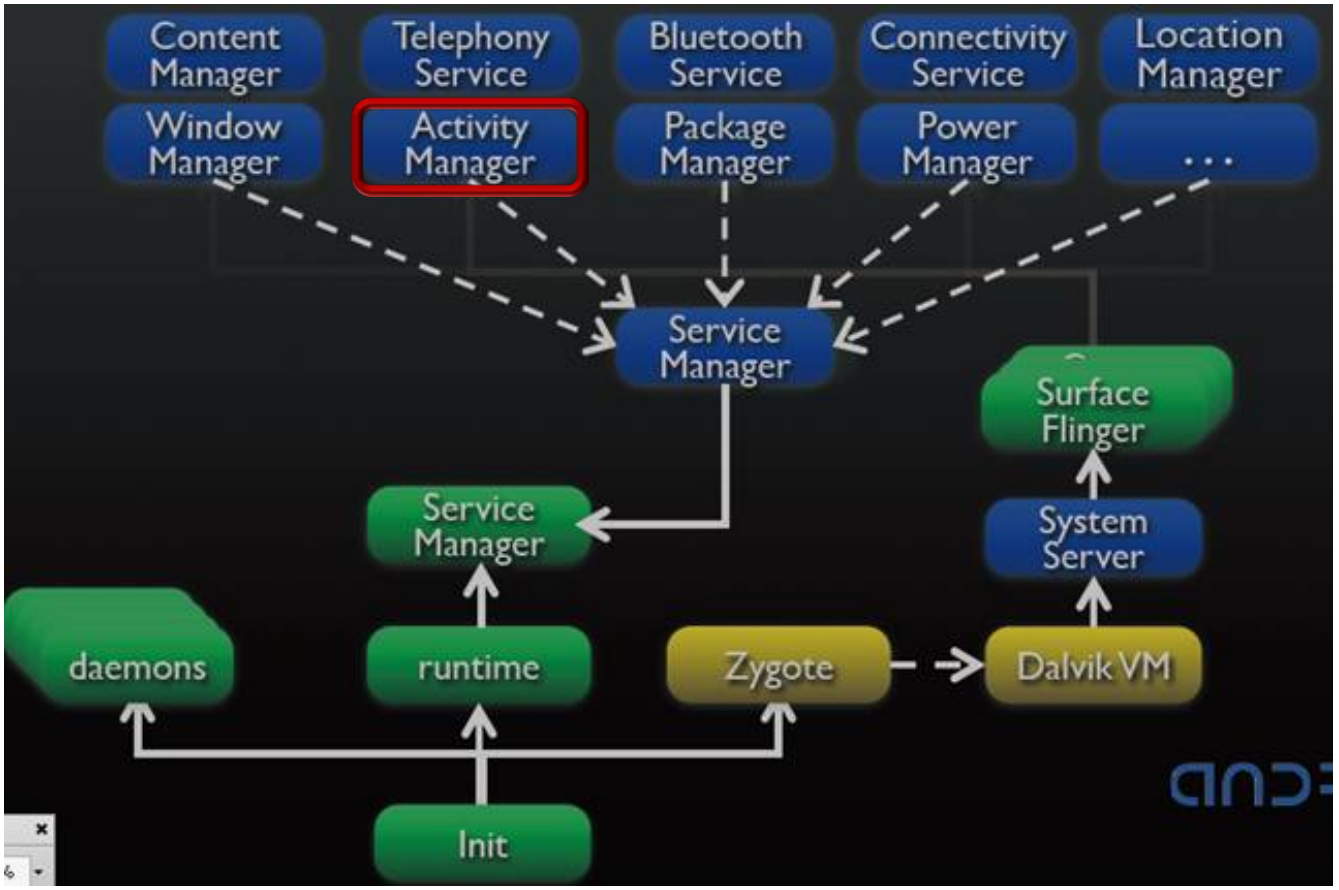
Managing the Activity Lifecycle

- Android communicates state changes to application by calling specific lifecycle methods



Managing the Activity Lifecycle

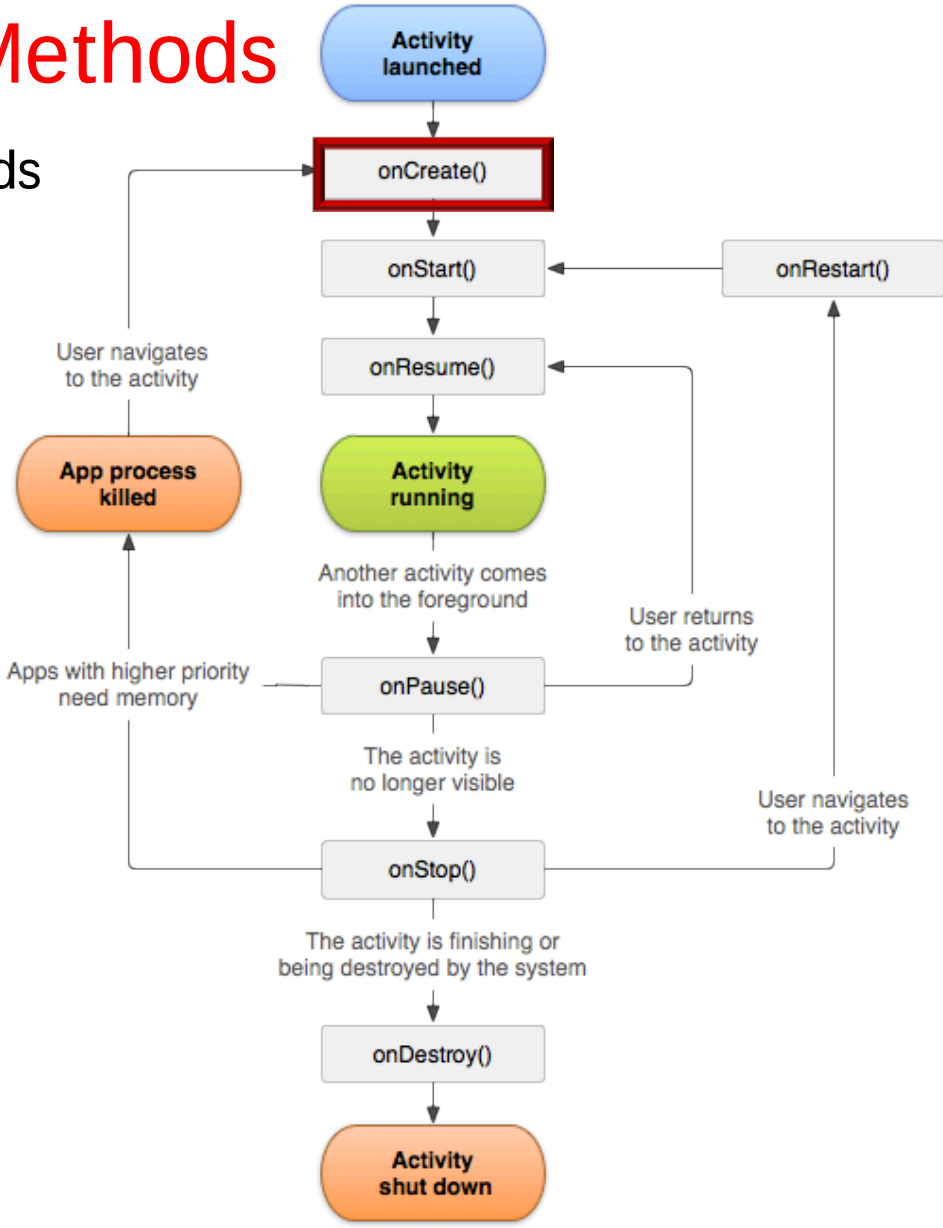
- Android communicates state changes to application by calling specific lifecycle methods
- The ActivityManager is the system service in Android that communicates these changes



developer.android.com/reference/android/app/ActivityManager.html has more

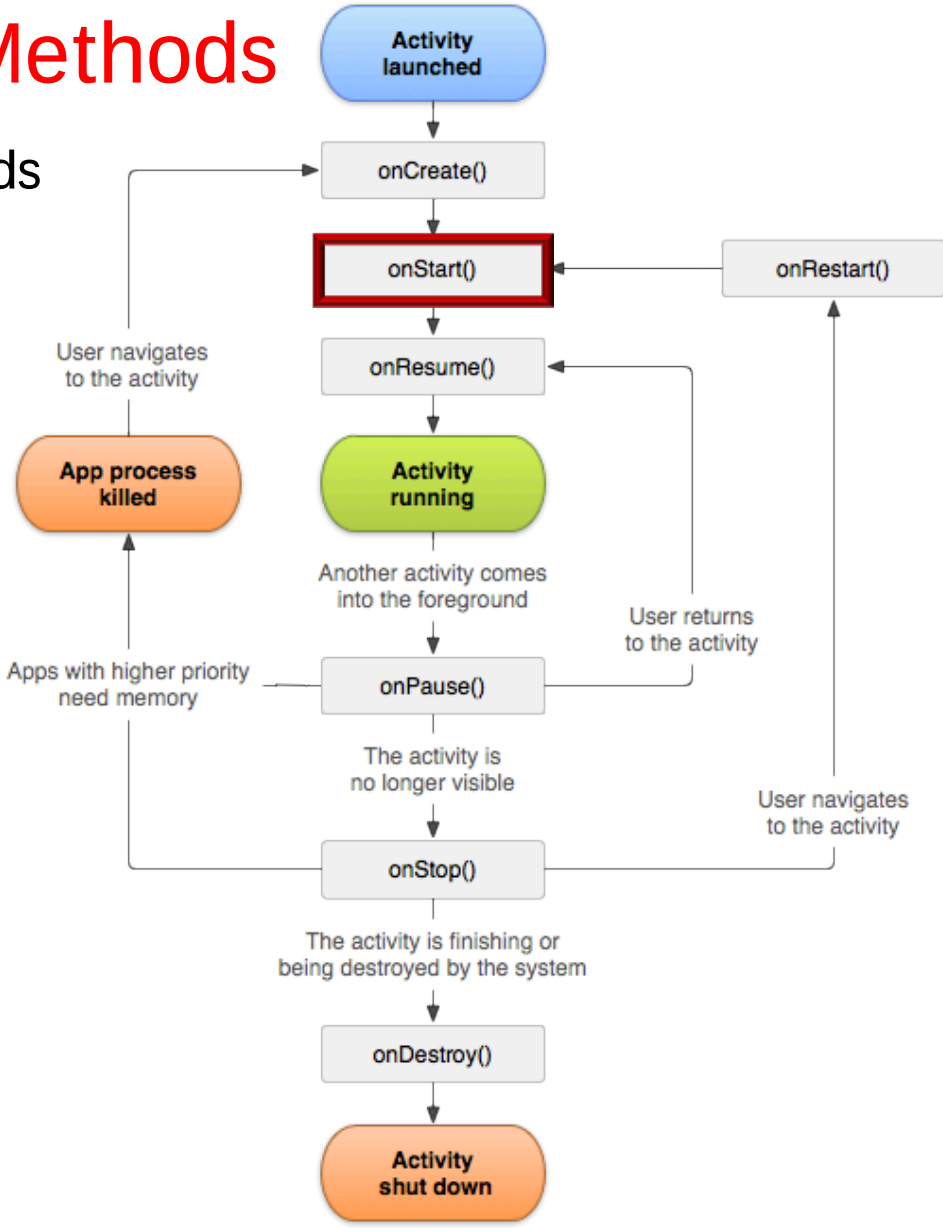
Activity Lifecycle Hook Methods

- The Android runtime calls hook methods on an Activity to control its lifecycle:
  - **onCreate()** – called to initialize an Activity when it is first created



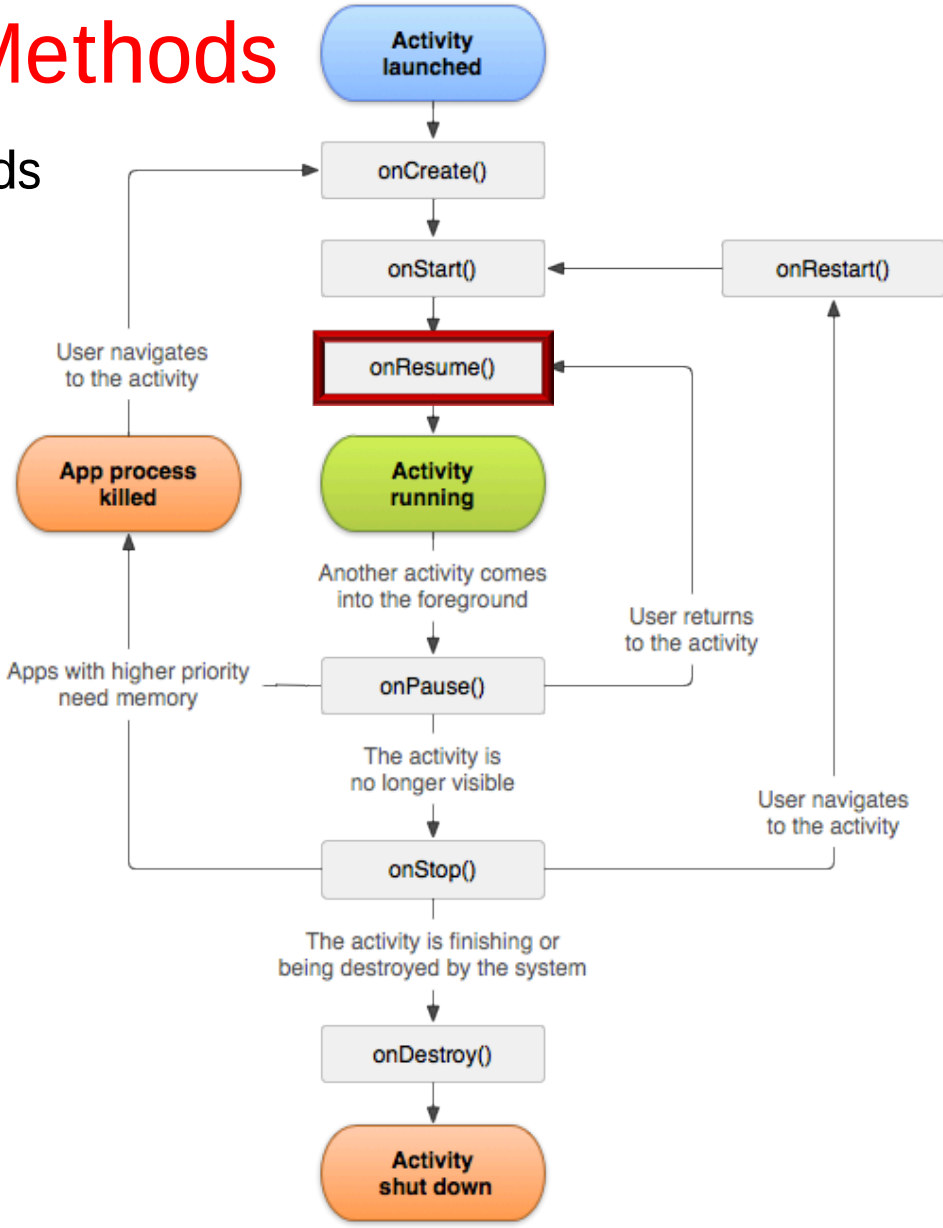
Activity Lifecycle Hook Methods

- The Android runtime calls hook methods on an Activity to control its lifecycle:
  - **onCreate()** – called to initialize an Activity when it is first created
  - **onStart()** – called when Activity is becoming visible to the user



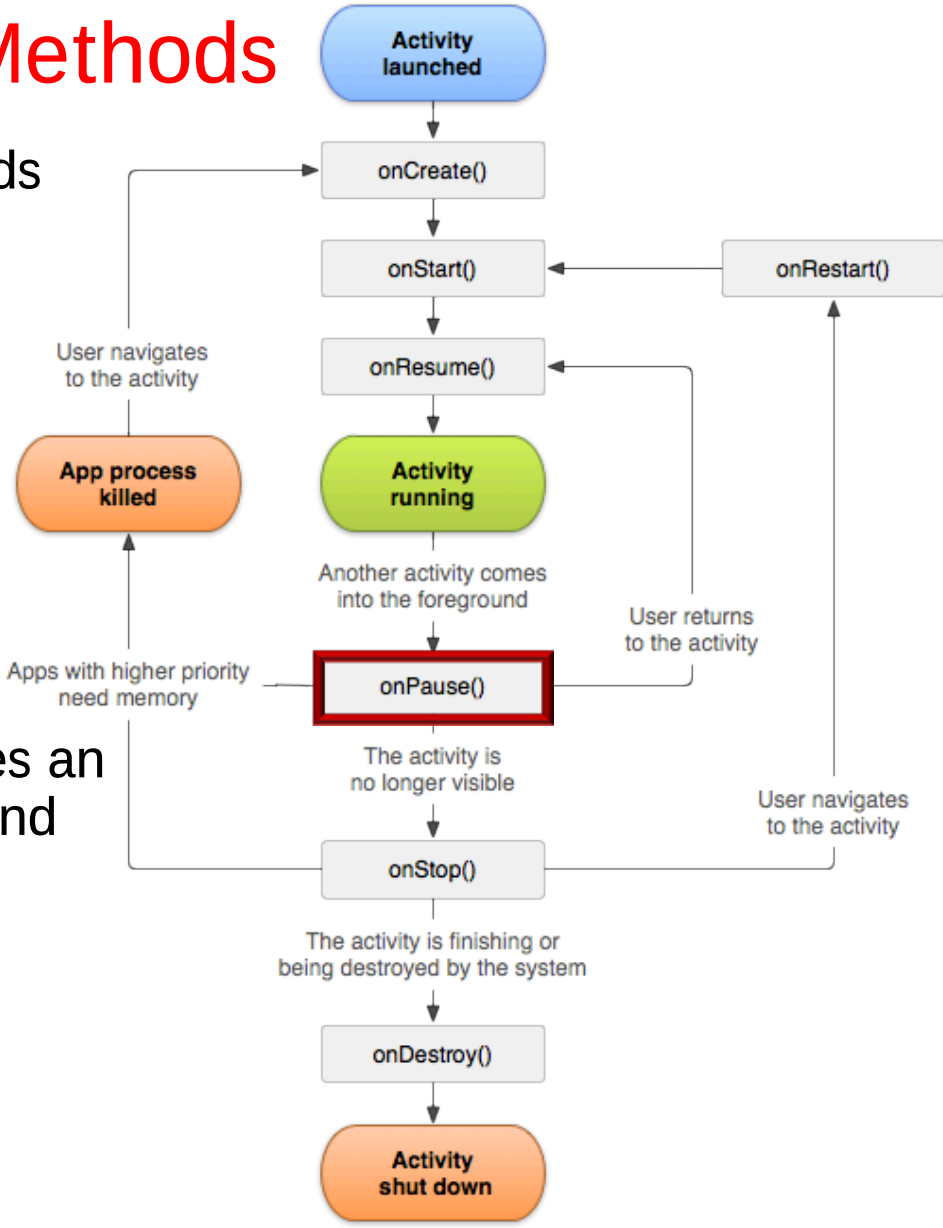
Activity Lifecycle Hook Methods

- The Android runtime calls hook methods on an Activity to control its lifecycle:
  - **onCreate()** – called to initialize an Activity when it is first created
  - **onStart()** – called when Activity is becoming visible to the user
  - **onResume()** – called when user returns to an Activity from another



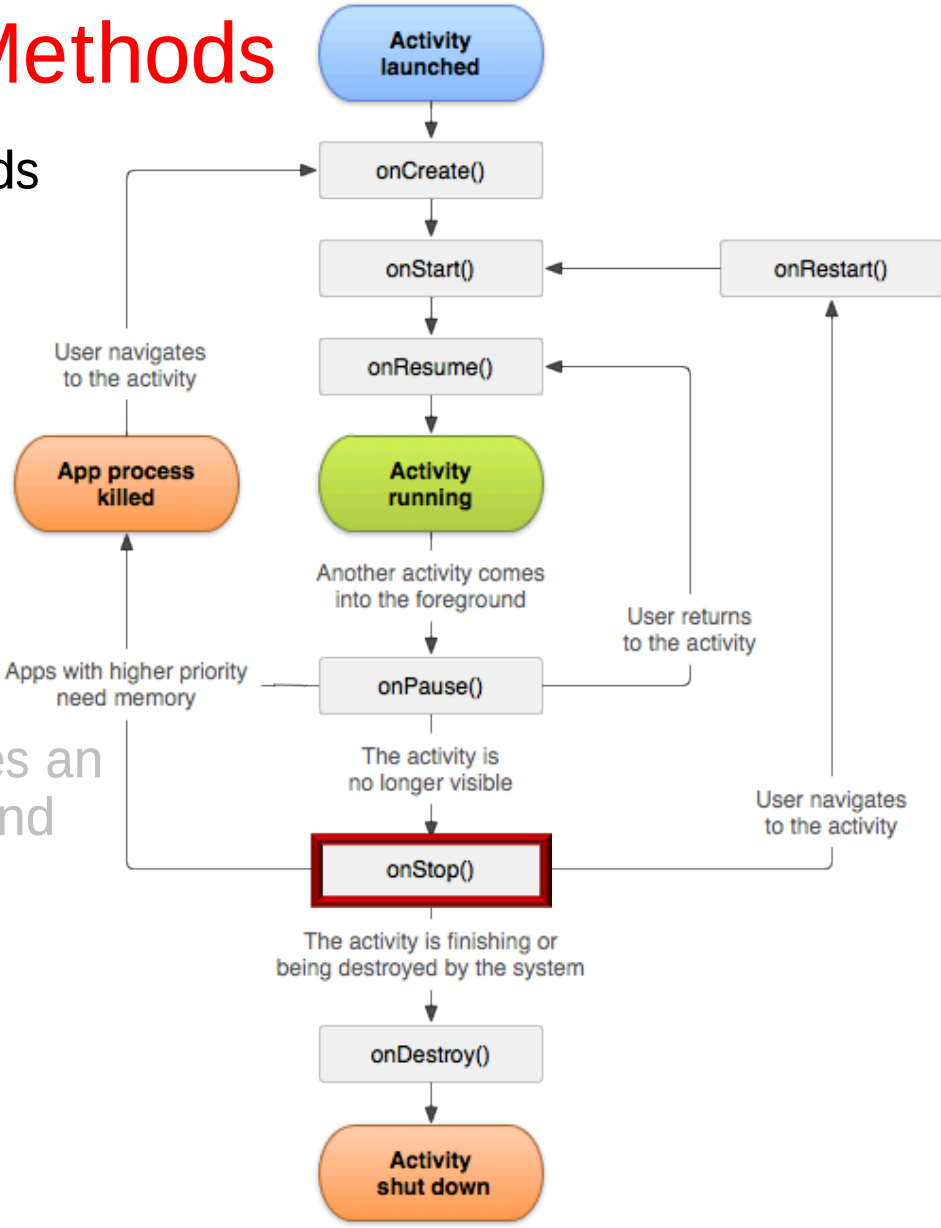
Activity Lifecycle Hook Methods

- The Android runtime calls hook methods on an Activity to control its lifecycle:
  - **onCreate()** – called to initialize an Activity when it is first created
  - **onStart()** – called when Activity is becoming visible to the user
  - **onResume()** – called when user returns to an Activity from another
  - **onPause()** – called when user leaves an Activity that's still visible in background



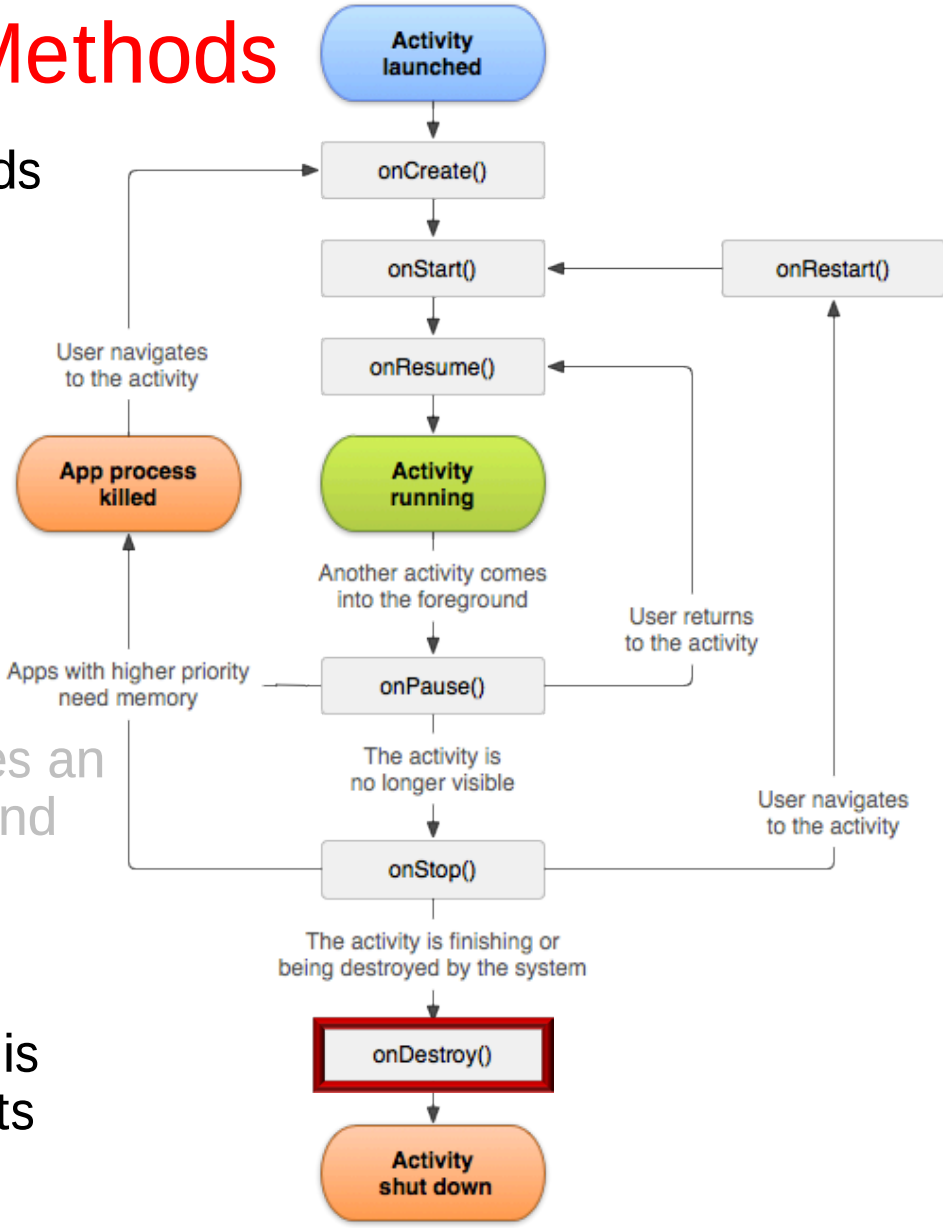
Activity Lifecycle Hook Methods

- The Android runtime calls hook methods on an Activity to control its lifecycle:
  - **onCreate()** – called to initialize an Activity when it is first created
  - **onStart()** – called when Activity is becoming visible to the user
  - **onResume()** – called when user returns to an Activity from another
  - **onPause()** – called when user leaves an Activity that's still visible in background
  - **onStop()** – called when user leaves an Activity for another



Activity Lifecycle Hook Methods

- The Android runtime calls hook methods on an Activity to control its lifecycle:
  - **onCreate()** – called to initialize an Activity when it is first created
  - **onStart()** – called when Activity is becoming visible to the user
  - **onResume()** – called when user returns to an Activity from another
  - **onPause()** – called when user leaves an Activity that's still visible in background
  - **onStop()** – called when user leaves an Activity for another
  - **onDestroy()** – called when Activity is being released & needs to clean up its allocated resources



See developer.android.com/reference/android/app/Activity.html for more info

Useful Helper Class for Activity Lifecycle Methods

```
public abstract class LifecycleLoggingActivity extends Activity {

  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d(getClass().getSimpleName(),
                "onCreate()");
    if (savedInstanceState == null)
      Log.d(getClass().getSimpleName(), "activity created anew");
    else
      Log.d(getClass().getSimpleName(), "activity restarted");
  }

  public void onStart() {
    super.onStart();
    Log.d(getClass().getSimpleName(), "onStart()");
  }
  ...
```

Inherit from Activity class

Automatically log lifecycle hook method calls

Note the “inversion of control” in the Android Activity framework

MapLocation App Example