Framework Examples: Part 2

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

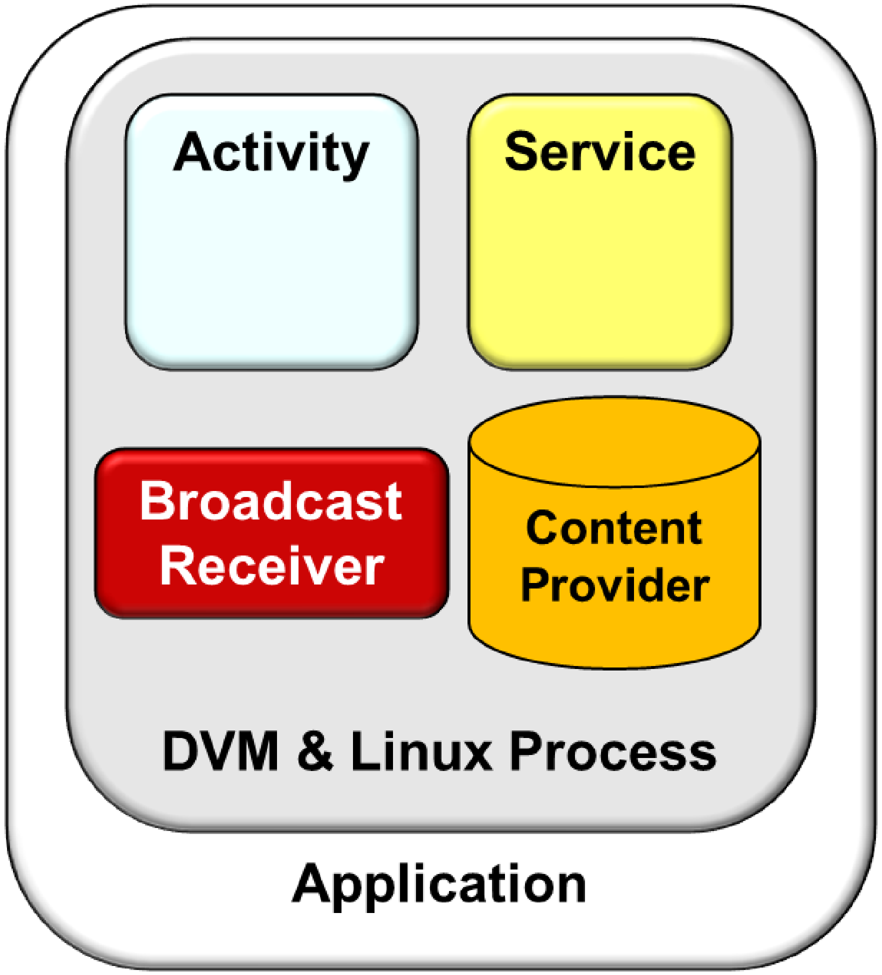
Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Topics Covered in this Part of the Module

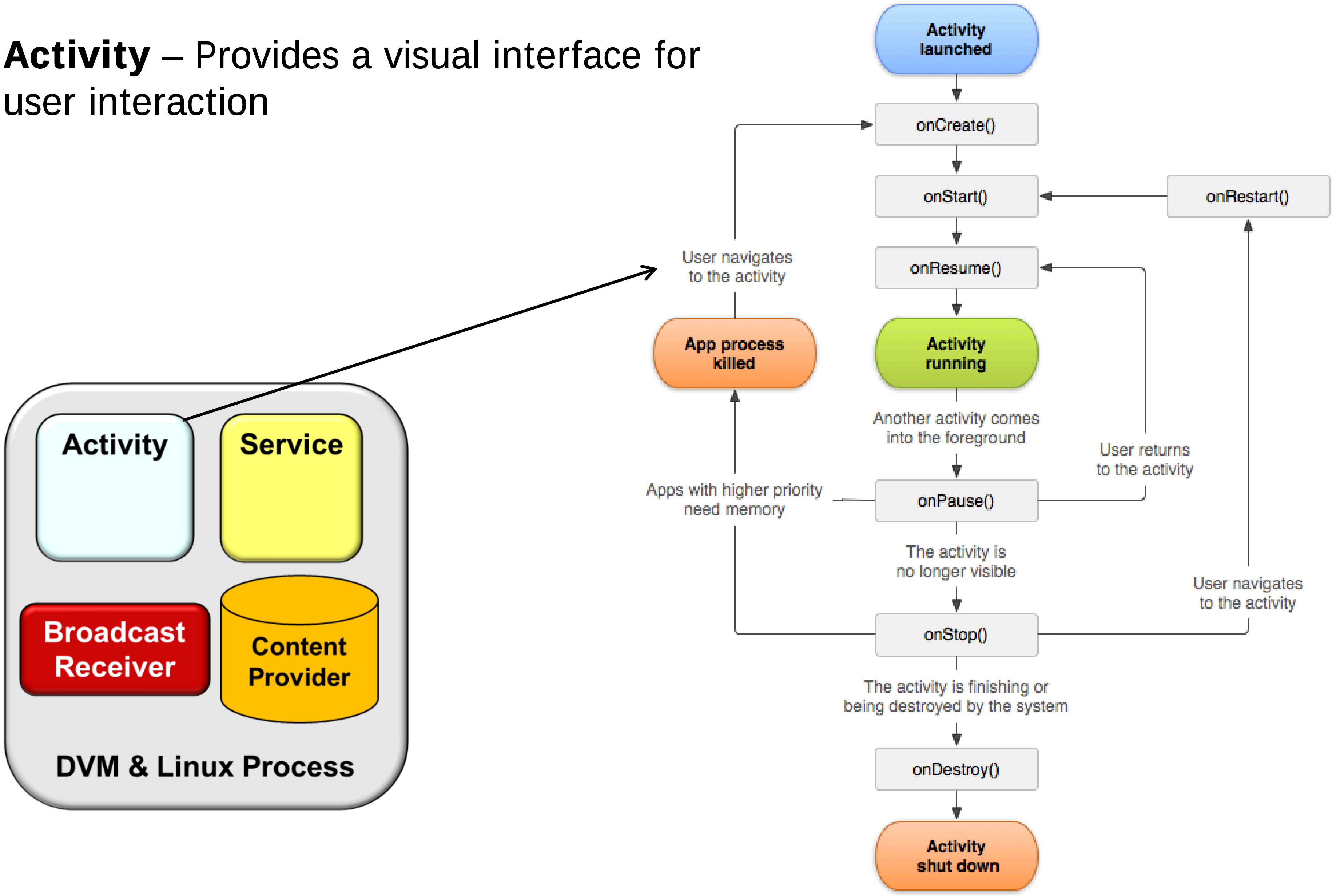
- Present *Scope, Commonality, & Variability* (SCV) analysis as a method for developing & applying software product-lines & frameworks
- Illustrate the application of SCV to the Android & ACE platforms
- Describe examples of Android framework components



We'll emphasize commonalities & variabilities in our discussion

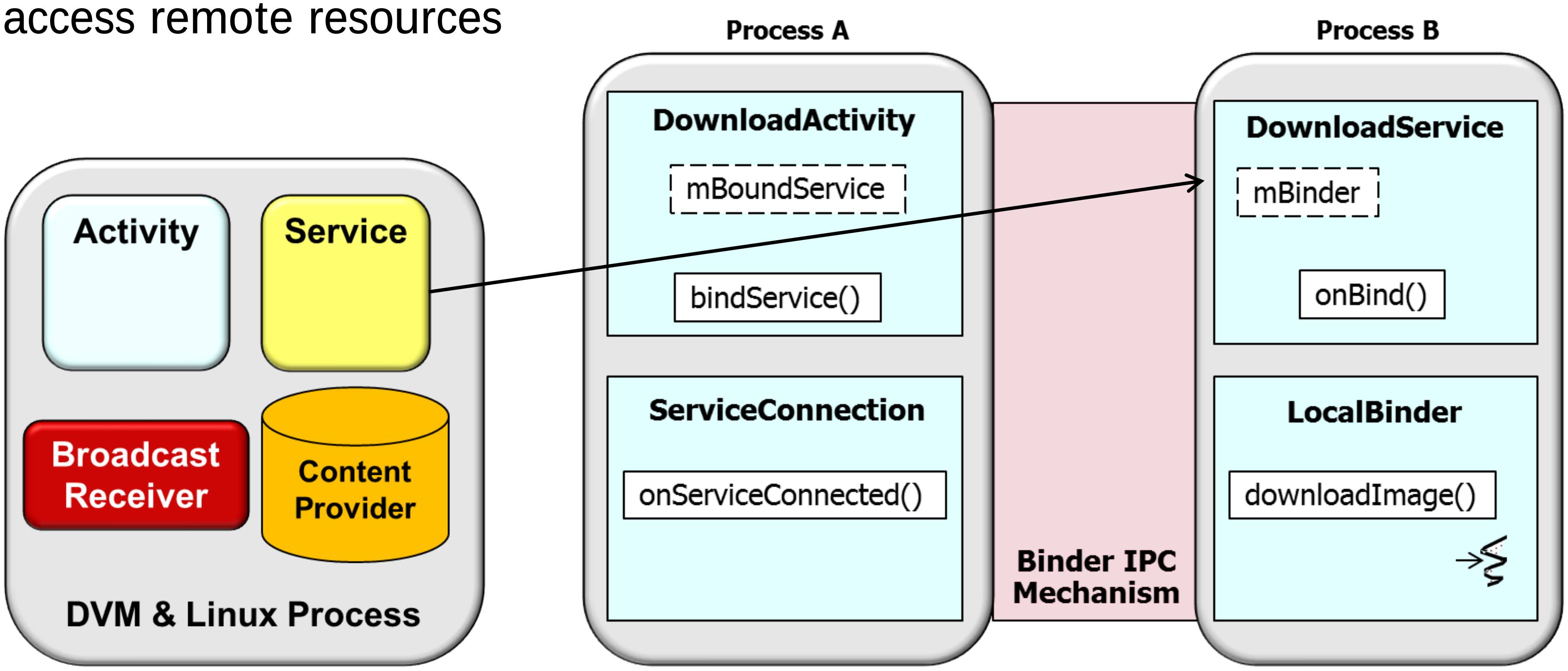
Example Android Framework Components

- **Activity** – Provides a visual interface for user interaction


Activity
Service
Broadcast Receiver
Content Provider
DVM & Linux Process
Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Example Android Framework Components

- **Activity** – Provides a visual interface for user interaction
- **Service** – Runs in background to perform long-running operations or to access remote resources

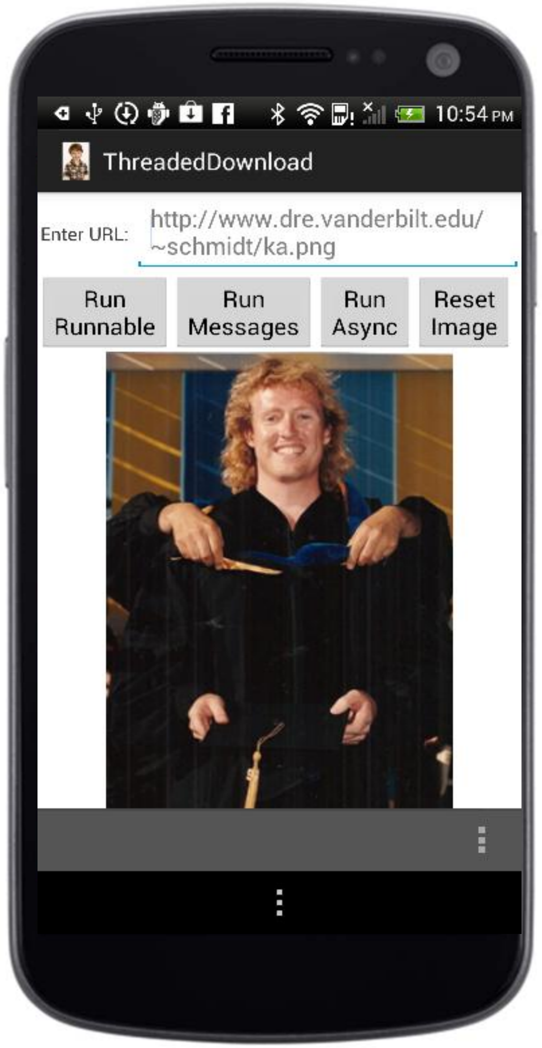


Recap of an Android Activity

- An Activity provides a visual interface for user interaction

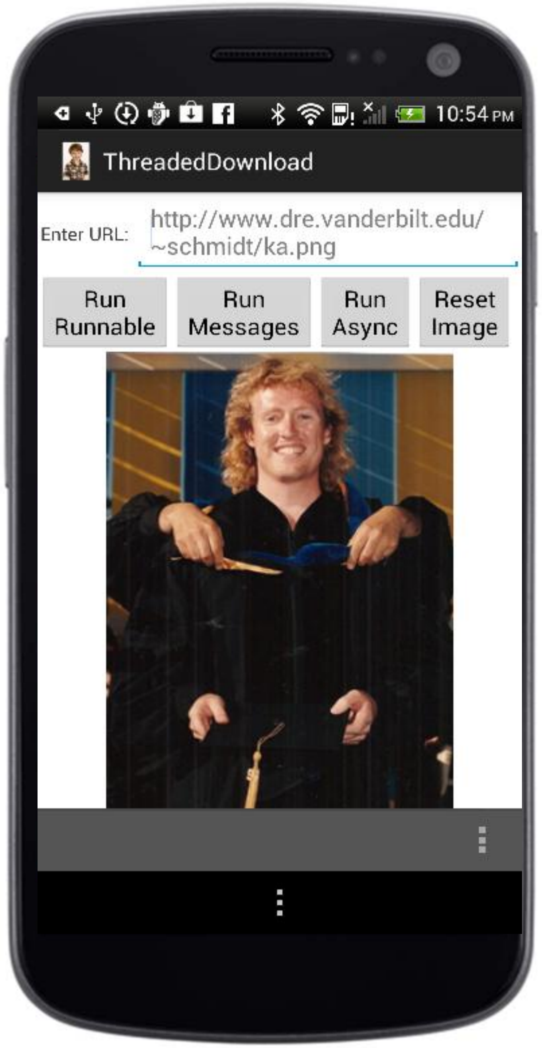
Recap of an Android Activity

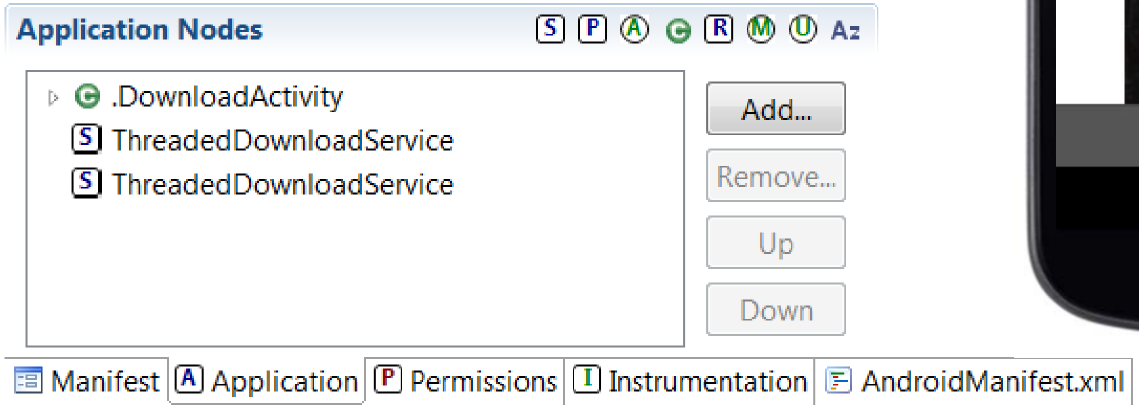
- An Activity provides a visual interface for user interaction
- Typically supports one thing a user can do, e.g.:
  - View an email message
  - Show a login screen
  - Download a file from a remote server



Recap of an Android Activity

- An Activity provides a visual interface for user interaction
- Typically supports one thing a user can do, e.g.:
  - View an email message
  - Show a login screen
  - Download a file from a remote server
- Applications can include one or more activities





.DownloadActivity
ThreadedDownloadService
ThreadedDownloadService

See developer.android.com/training/basics/activity-lifecycle/index.html for more

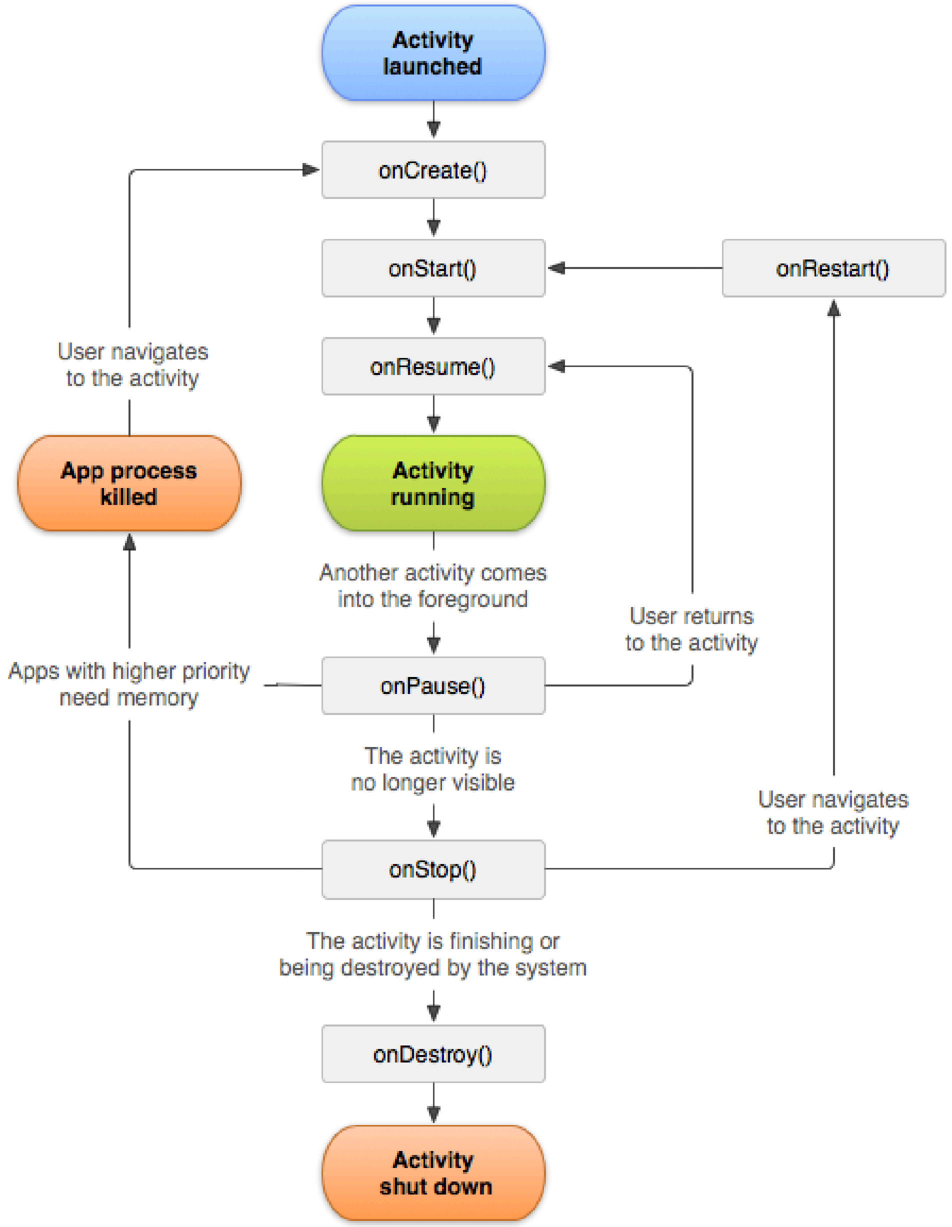
Implementing an Activity

- Implementing an Activity involves several steps
  - e.g., inherit from Activity class, override lifecycle hook methods, include Activity in the config file AndroidManifest.xml, etc.

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

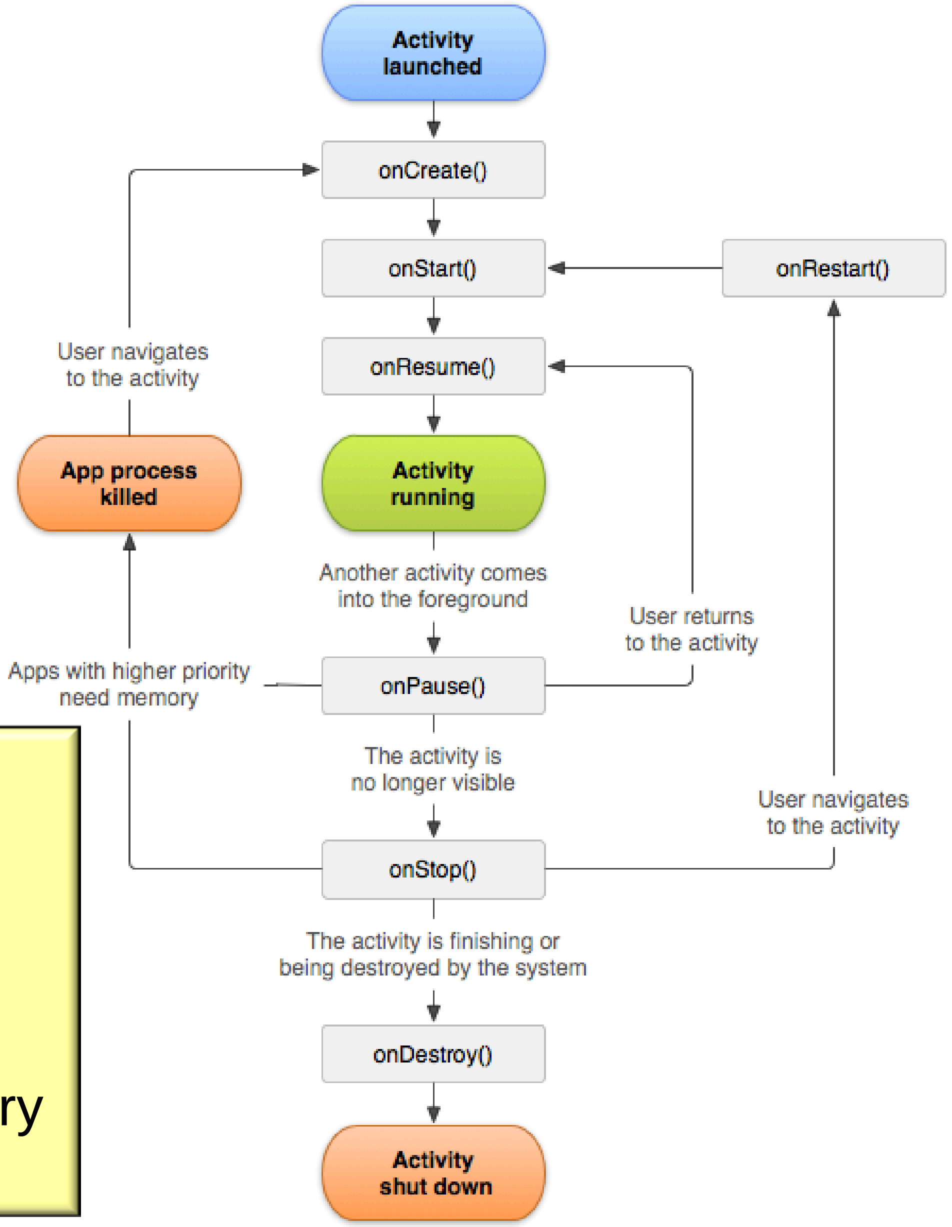
Implementing an Activity

- Implementing an Activity involves several steps
  - e.g., inherit from Activity class, override lifecycle hook methods, include Activity in the config file AndroidManifest.xml, etc.
- Android communicates state changes to an Activity by calling its lifecycle hook methods


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Implementing an Activity

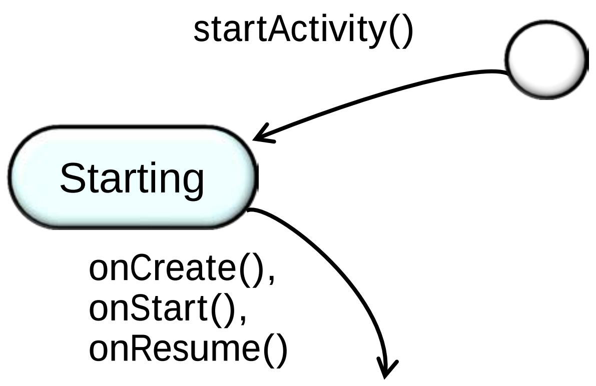
- Implementing an Activity involves several steps
  - e.g., inherit from Activity class, override lifecycle hook methods, include Activity in the config file AndroidManifest.xml, etc.
- Android communicates state changes to an Activity by calling its lifecycle hook methods


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

- **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

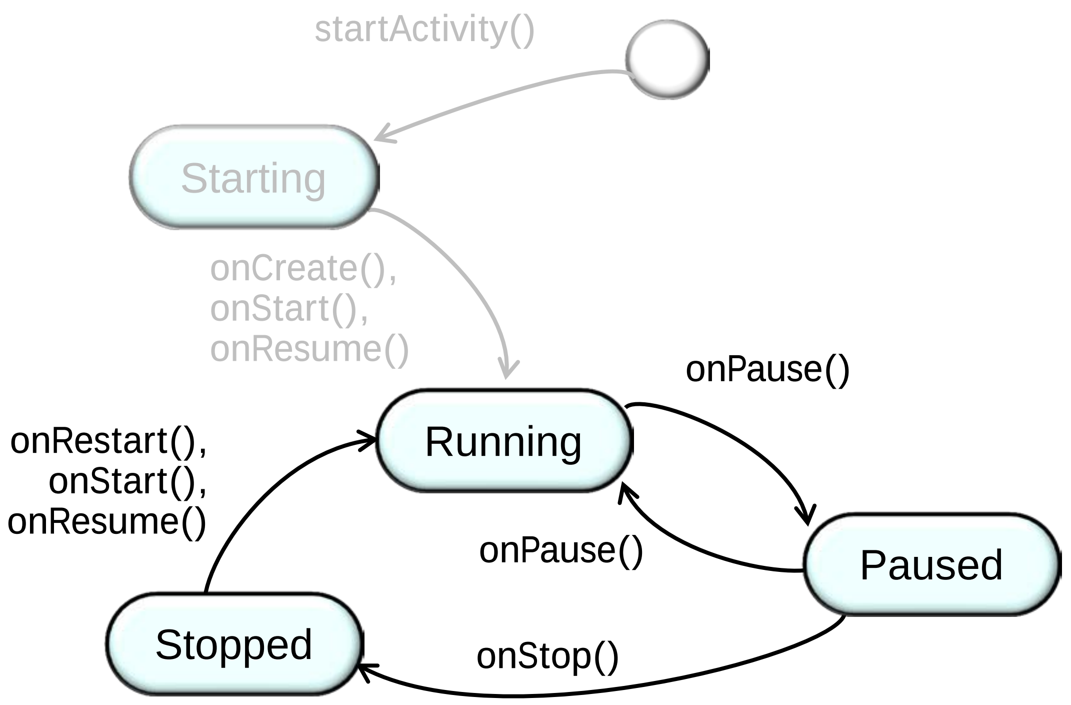
Activity Lifecycle States

- **Activity starting** – Initialization steps



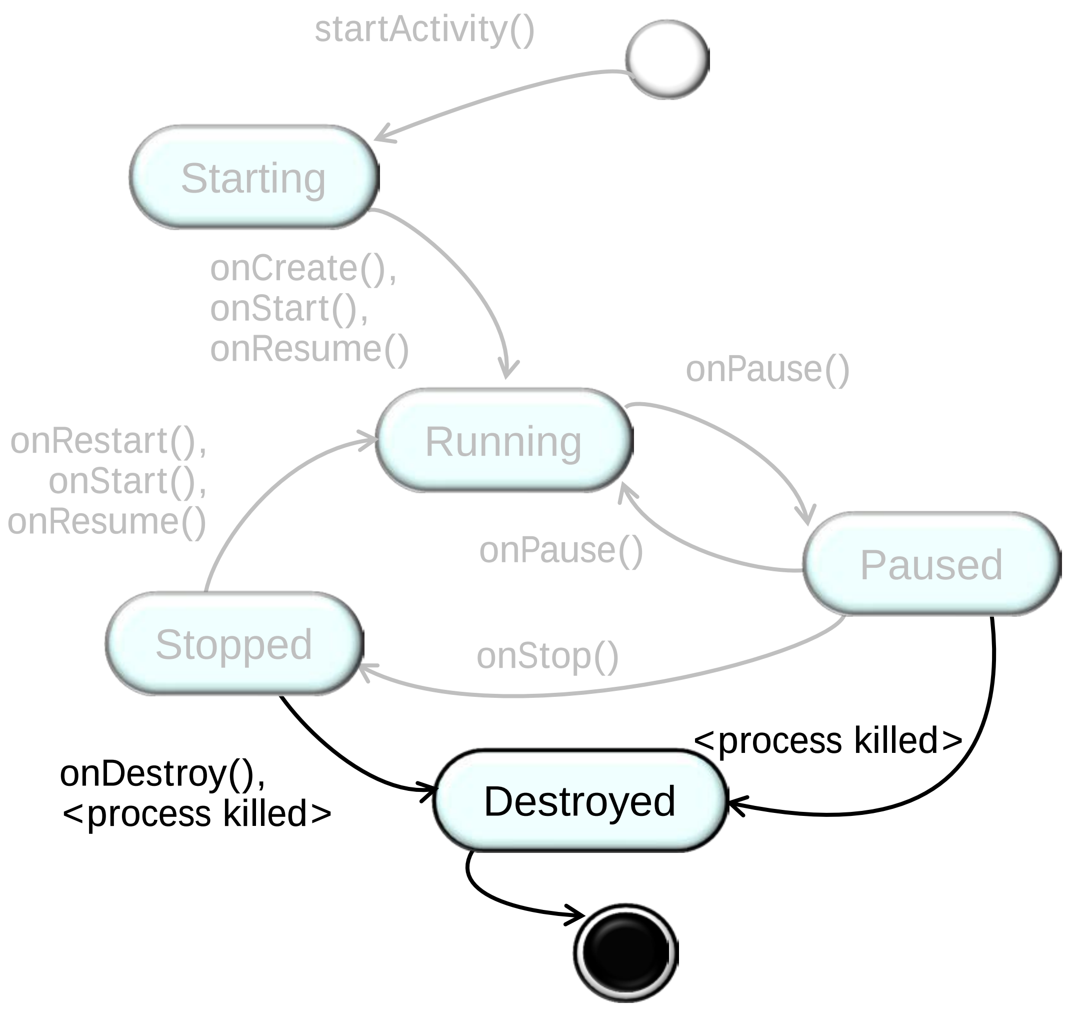
Activity Lifecycle States

- **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

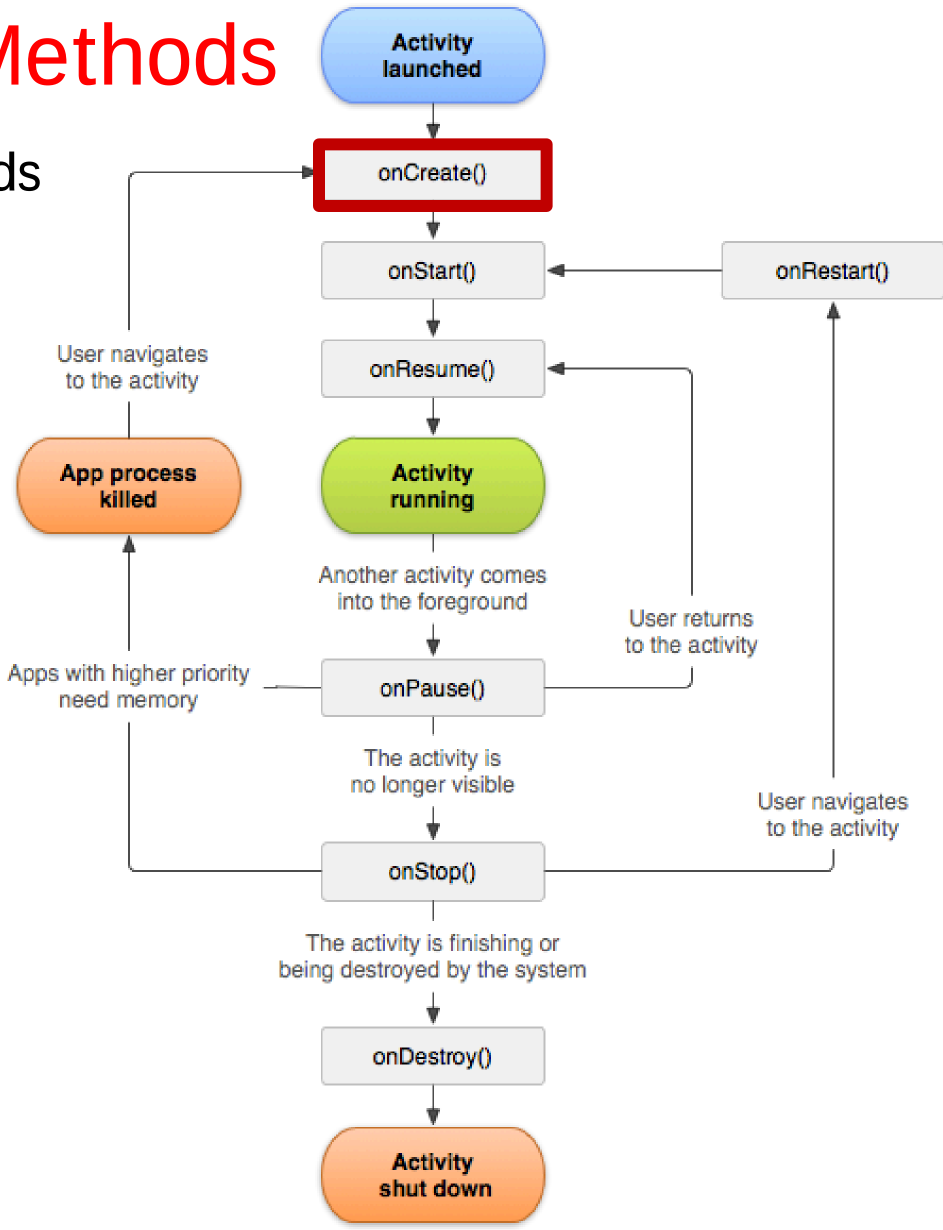
- **Activity starting** – Initialization steps
- **Activity running**
  - *Running* – visible, has focus
  - *Paused* – visible, does not have focus, can be terminated
  - *Stopped* – not visible, does not have focus, can be terminated
- **Activity shut down** – Voluntarily finished or involuntarily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


startActivity()
Starting
onCreate(), onStart(), onResume()
Running
onPause()
Paused
onRestart(), onStart(), onResume()
onPause()
Stopped
onStop()
<process killed>
onDestroy(), <process killed>
Destroyed

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

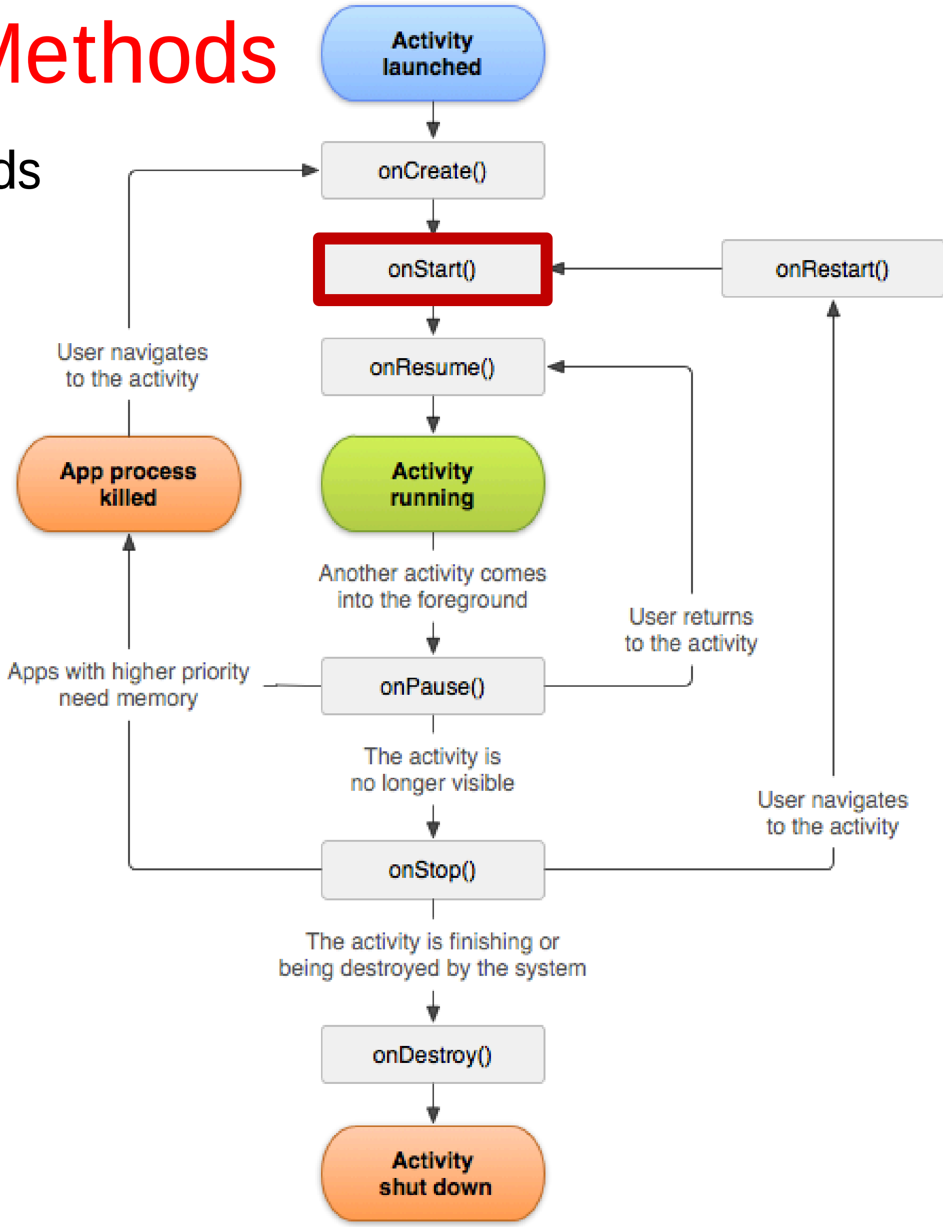
Activity Lifecycle Hook Methods

- 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 launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

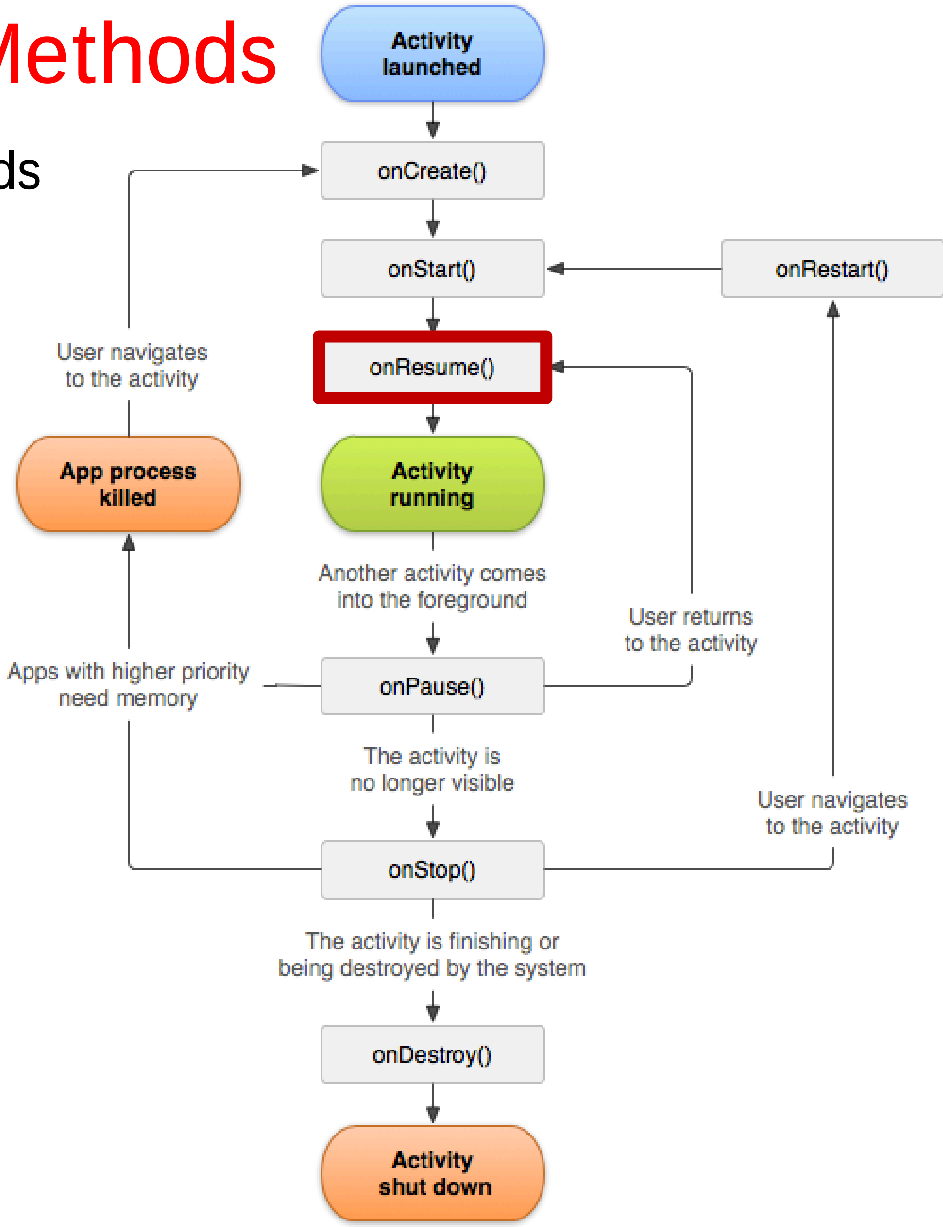
Activity Lifecycle Hook Methods

- 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 launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

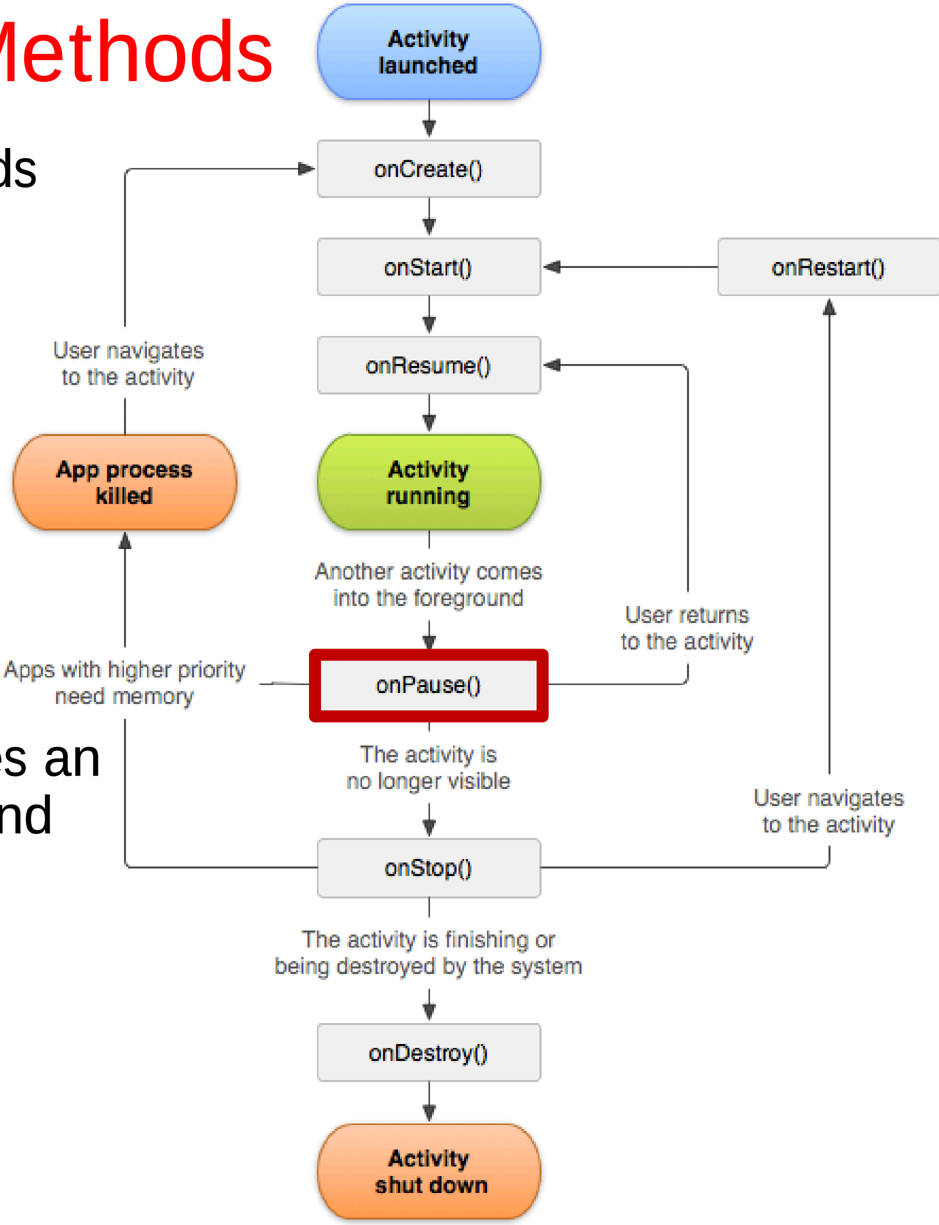
Activity Lifecycle Hook Methods

- 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 launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

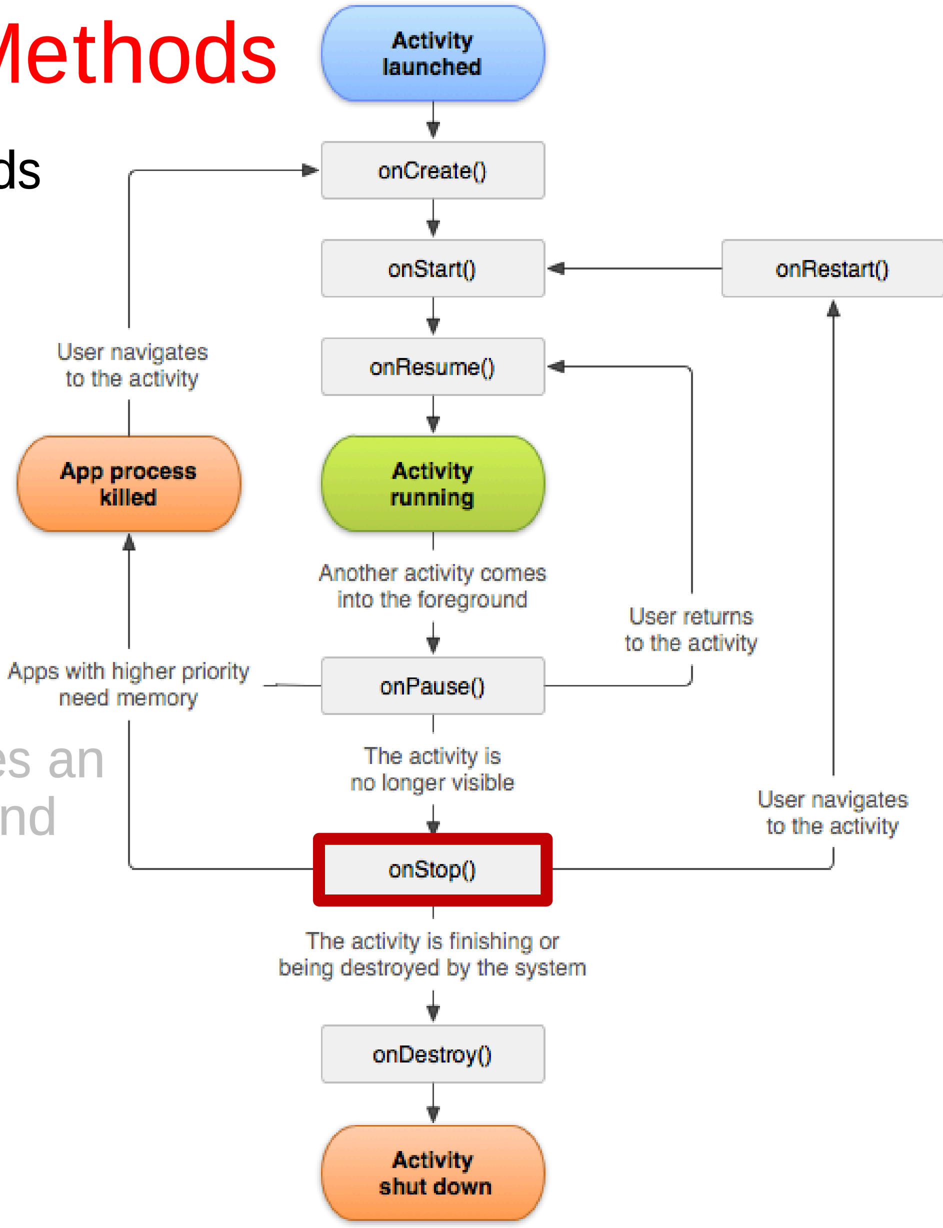
Activity Lifecycle Hook Methods

- 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 launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

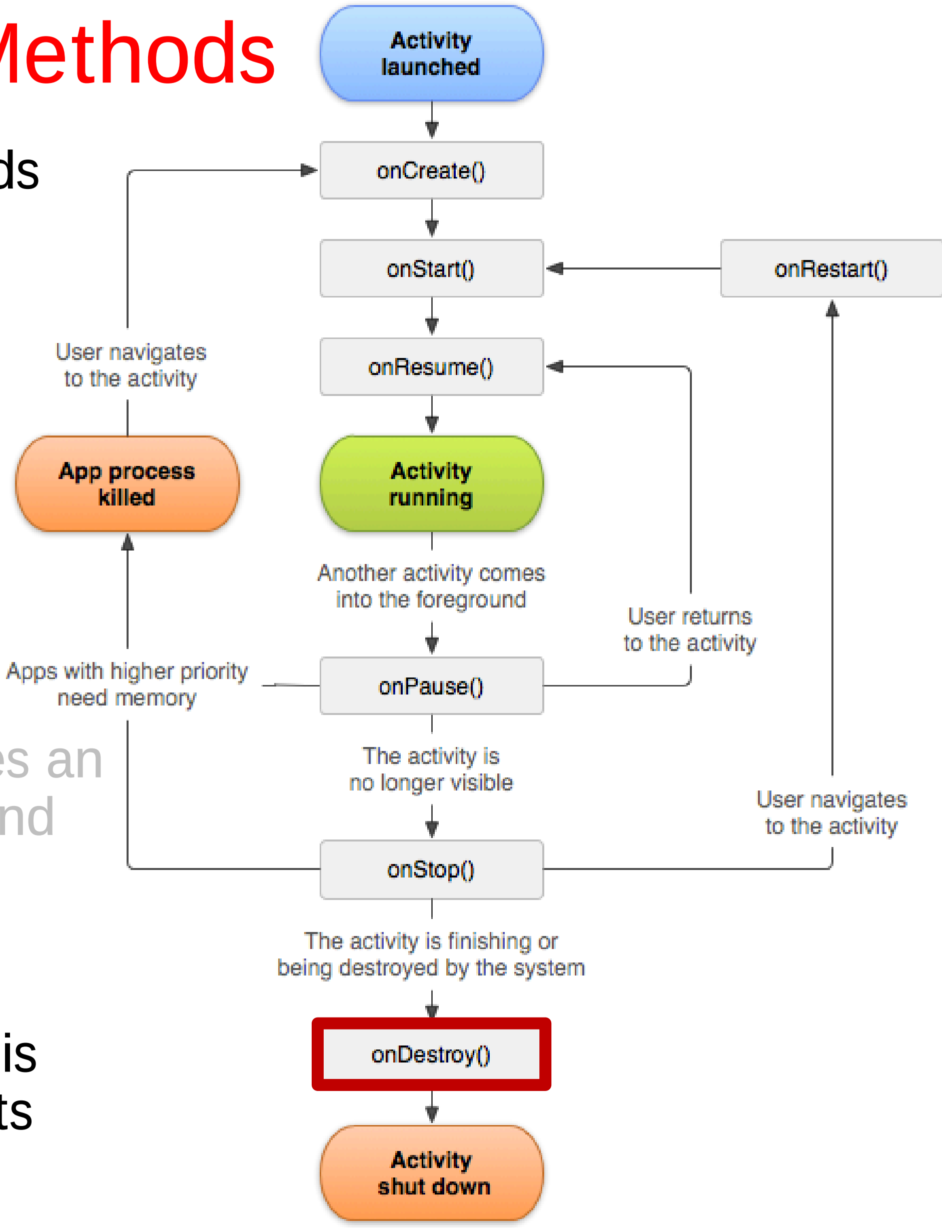
Activity Lifecycle Hook Methods

- 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 launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

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


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

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 {
```

Inherit from Activity class

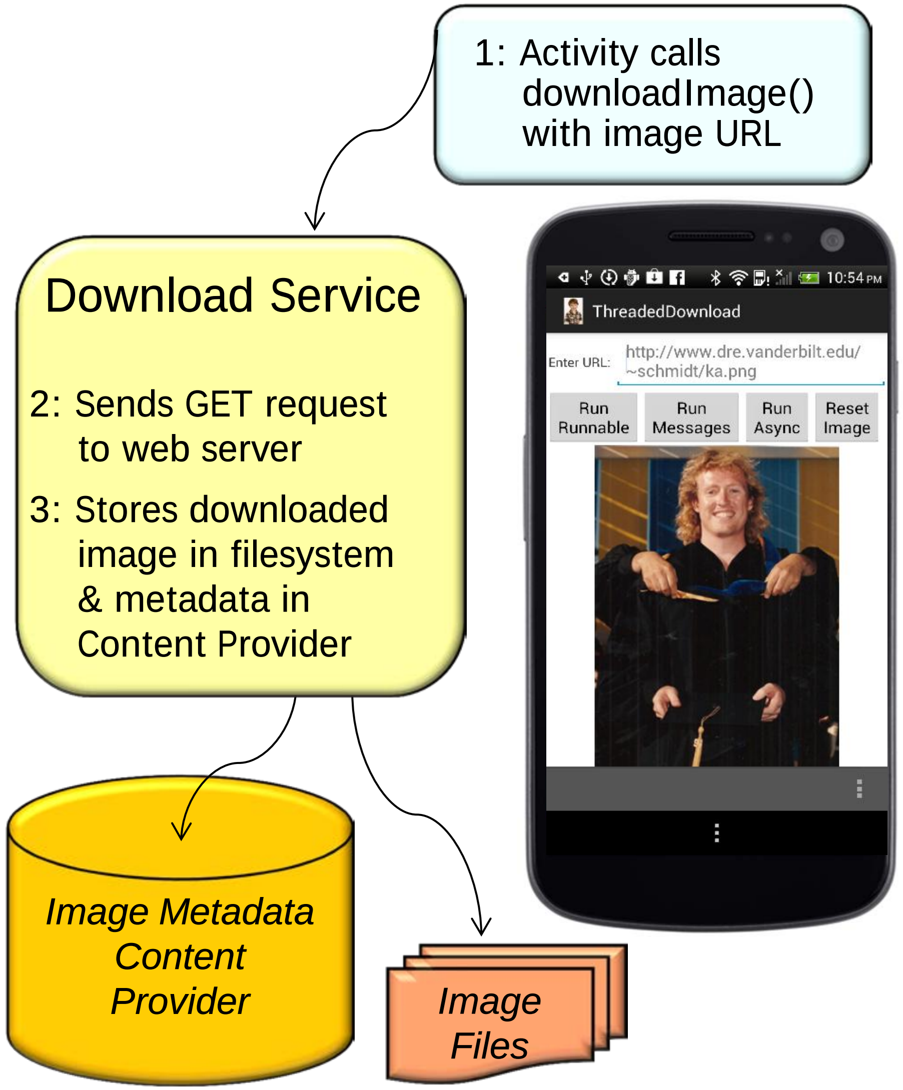
```
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d(getClass().getSimpleName(),
               "onCreate()");
    if (savedInstanceState == null)
      Log.d(getClass().getSimpleName(), "activity created anew");
    else
      Log.d(getClass().getSimpleName(),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

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

Automatically log lifecycle hook method calls

Note the "inversion of control" in the Android Activity framework

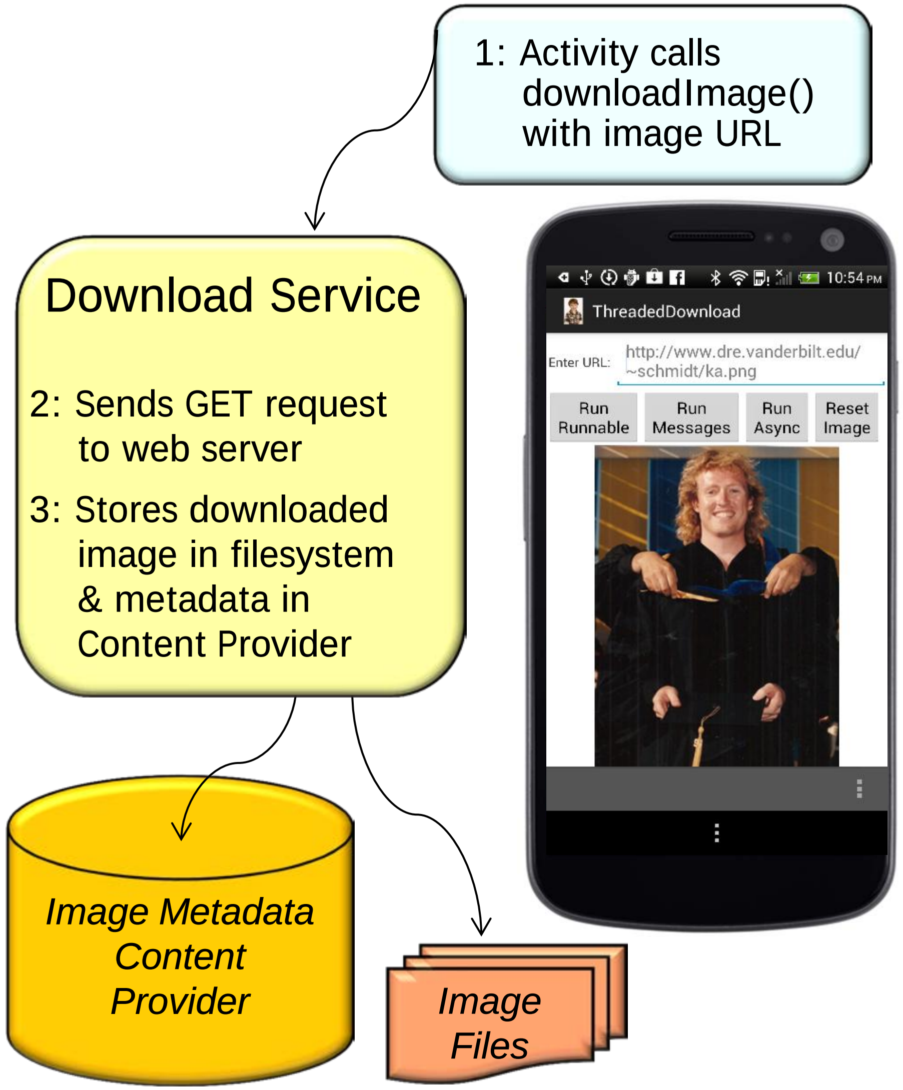
Recap of an Android Service

- A Service is an app component that can perform long-running operations in the background & does not provide direct access to the user interface
  - e.g., a service might handle network transactions, play music, perform file I/O, interact with a content provider, or run periodic tasks


1: Activity calls downloadImage() with image URL
Download Service
2: Sends GET request to web server
3: Stores downloaded image in filesystem & metadata in Content Provider
Image Metadata Content Provider
Image Files
10:54 PM
ThreadedDownload
Enter URL:
http://www.dre.vanderbilt.edu/~schmidt/ka.png
Run Runnable
Run Messages
Run Async
Reset Image

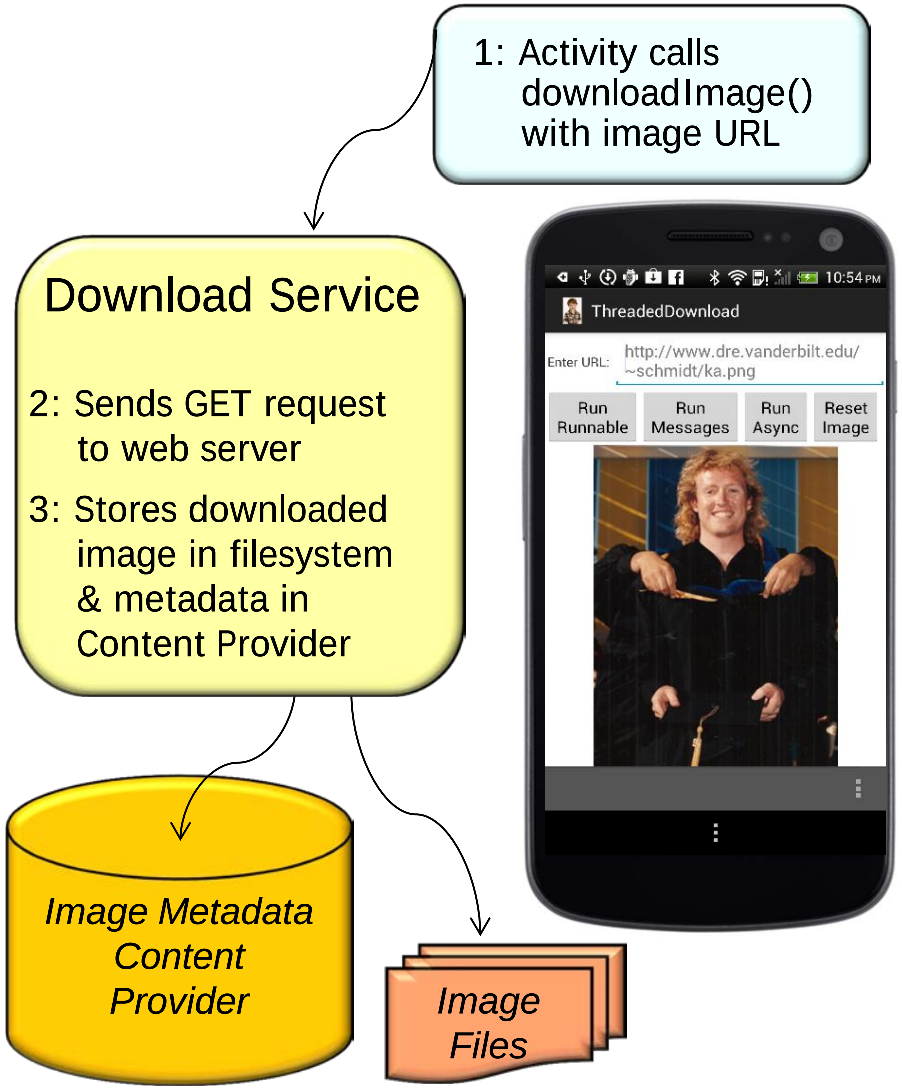
Recap of an Android Service

- A Service is an app component that can perform long-running operations in the background & does not provide direct access to the user interface
  - e.g., a service might handle network transactions, play music, perform file I/O, interact with a content provider, or run periodic tasks
- Another app component can start a service & it will continue to run in the background even if the user switches to another app/activity


1: Activity calls downloadImage() with image URL
Download Service
2: Sends GET request to web server
3: Stores downloaded image in filesystem & metadata in Content Provider
Image Metadata Content Provider
Image Files
10:54 PM
ThreadedDownload
Enter URL: http://www.dre.vanderbilt.edu/~schmidt/ka.png
Run Runnable
Run Messages
Run Async
Reset Image

Recap of an Android Service

- A Service is an app component that can perform long-running operations in the background & does not provide direct access to the user interface
  - e.g., a service might handle network transactions, play music, perform file I/O, interact with a content provider, or run periodic tasks
- Another app component can start a service & it will continue to run in the background even if the user switches to another app/activity
- Components can also bind to services to interact with them & perform local or remote IPC


1: Activity calls downloadImage() with image URL
Download Service
2: Sends GET request to web server
3: Stores downloaded image in filesystem & metadata in Content Provider
Image Metadata Content Provider
Image Files
10:54 PM
ThreadedDownload
Enter URL: http://www.dre.vanderbilt.edu/~schmidt/ka.png
Run Runnable
Run Messages
Run Async
Reset Image

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

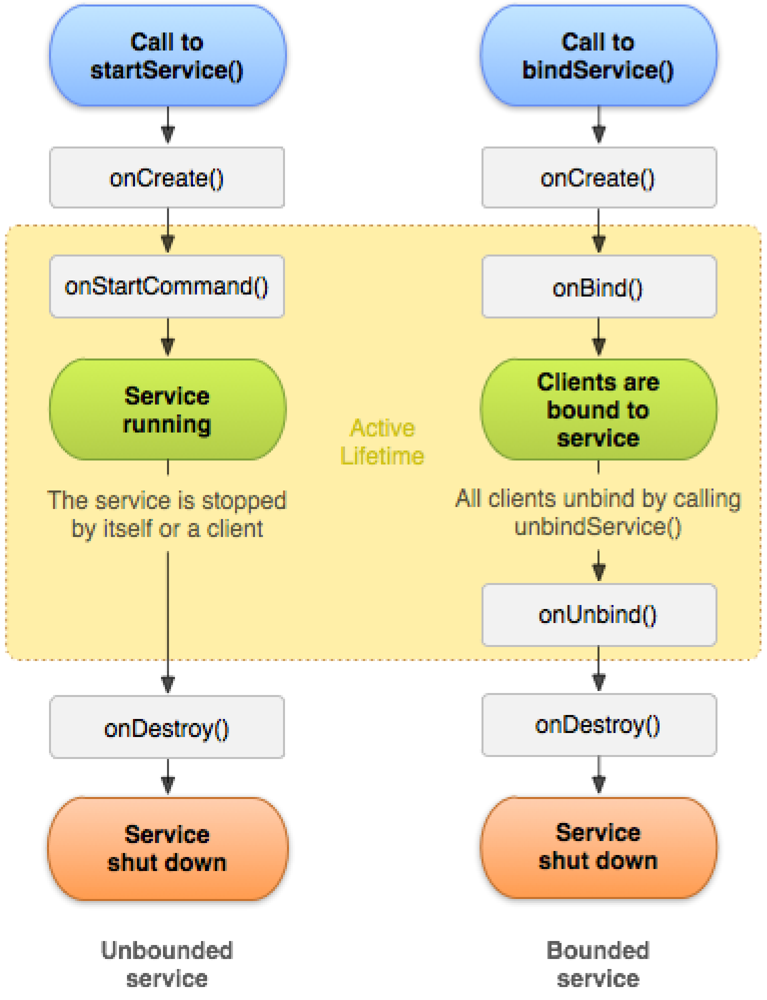
Implementing a Service

- Implementing a Service is similar to implementing an Activity
    - e.g., inherit from Android Service class, override lifecycle methods, include Service in the config file AndroidManifest.xml, etc.

```
public class Service extends
                ... {
   public void onCreate();
   public int onStartCommand
     (Intent intent,
      int flags, int startId);
   public abstract IBinder
      onBind(Intent intent);
   public boolean
      onUnbind(Intent intent);
   protected void onDestroy();
   ...
}
```

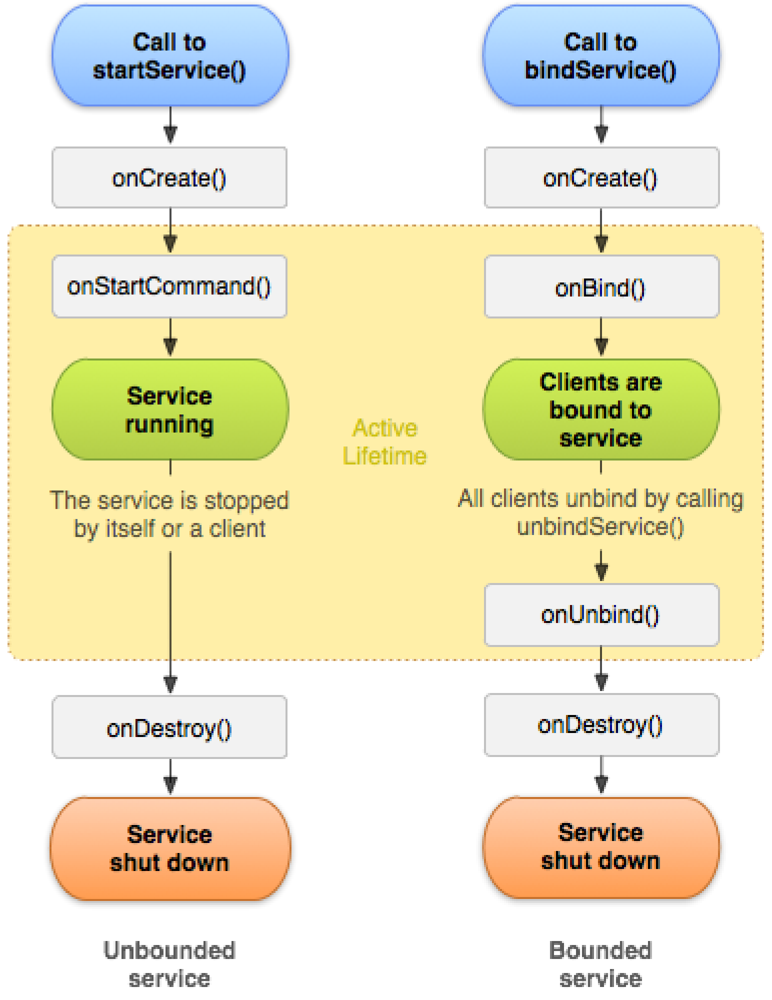
Implementing a Service

- Implementing a Service is similar to implementing an Activity
  - e.g., inherit from Android Service class, override lifecycle methods, include Service in the config file AndroidManifest.xml, etc.
- Android communicates state changes to a Service by calling its lifecycle hook methods



Implementing a Service

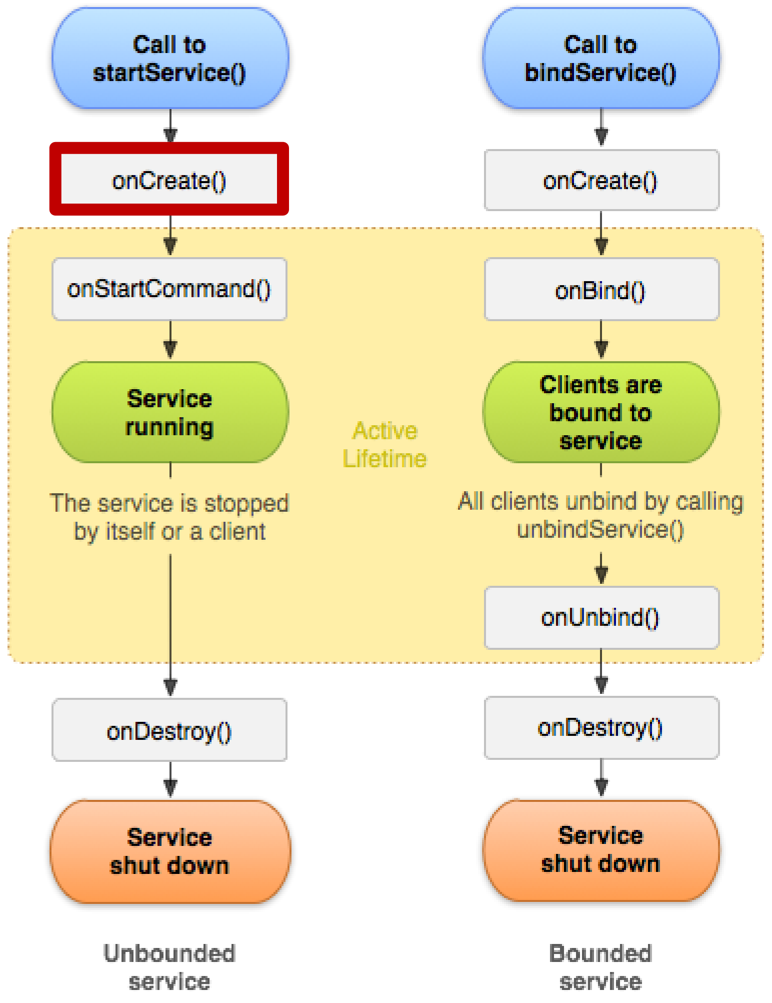
- Implementing a Service is similar to implementing an Activity
  - e.g., inherit from Android Service class, override lifecycle methods, include Service in the config file AndroidManifest.xml, etc.
- Android communicates state changes to a Service by calling its lifecycle hook methods



- **Commonality**: Provides common interface for performing long-running operations that don’t interact directly with the user interface
- **Variability**: Subclasses can override lifecycle hook methods to perform necessary initialization for *Started* & *Bound* Services

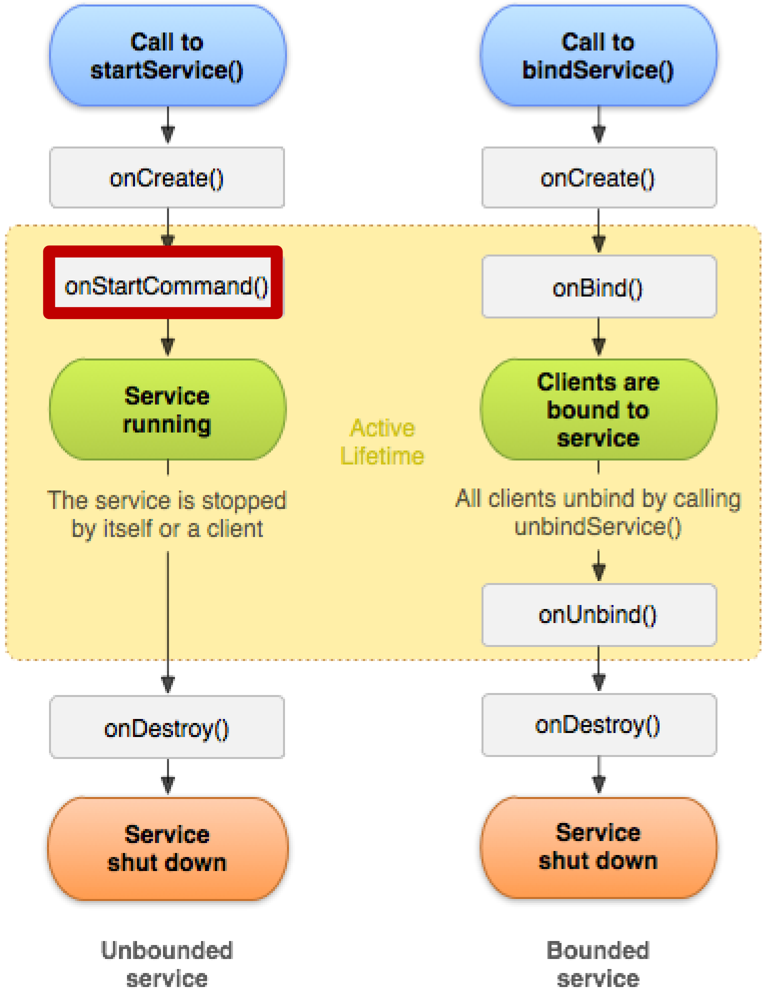
Service Lifecycle Hook Methods

- Services lifecycle methods include
  - **onCreate()** – called when Service process is created, by any means



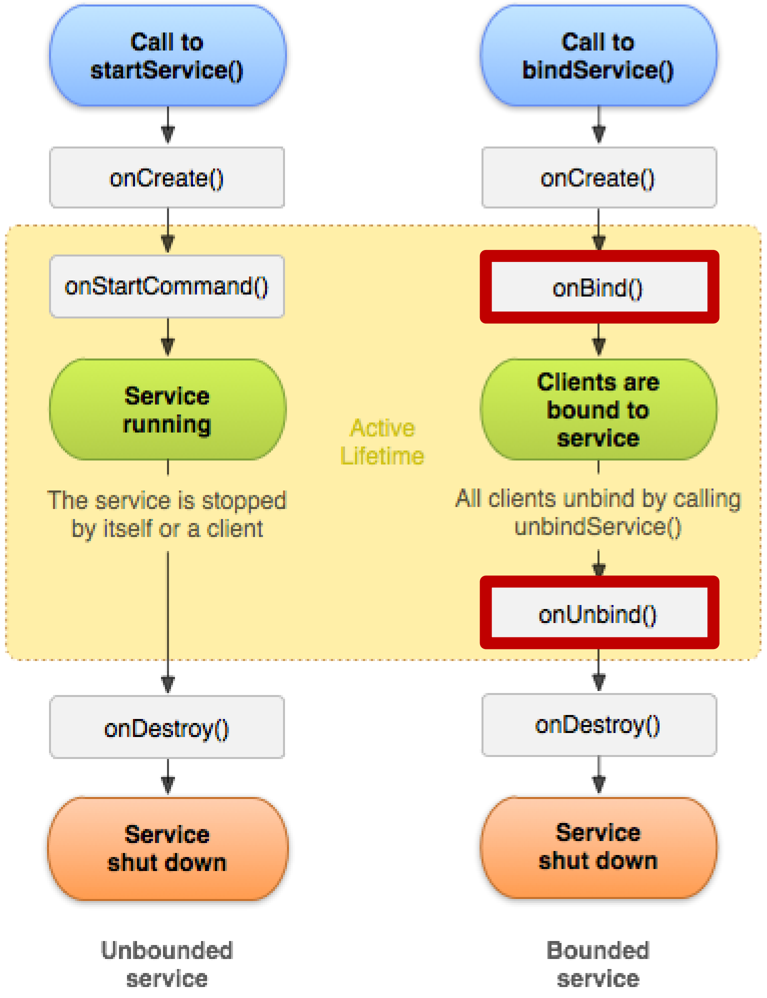
Service Lifecycle Hook Methods

- Services lifecycle methods include
  - **onCreate()** – called when Service process is created, by any means
  - **onStartCommand()** – called each time Service is sent a command via startService()



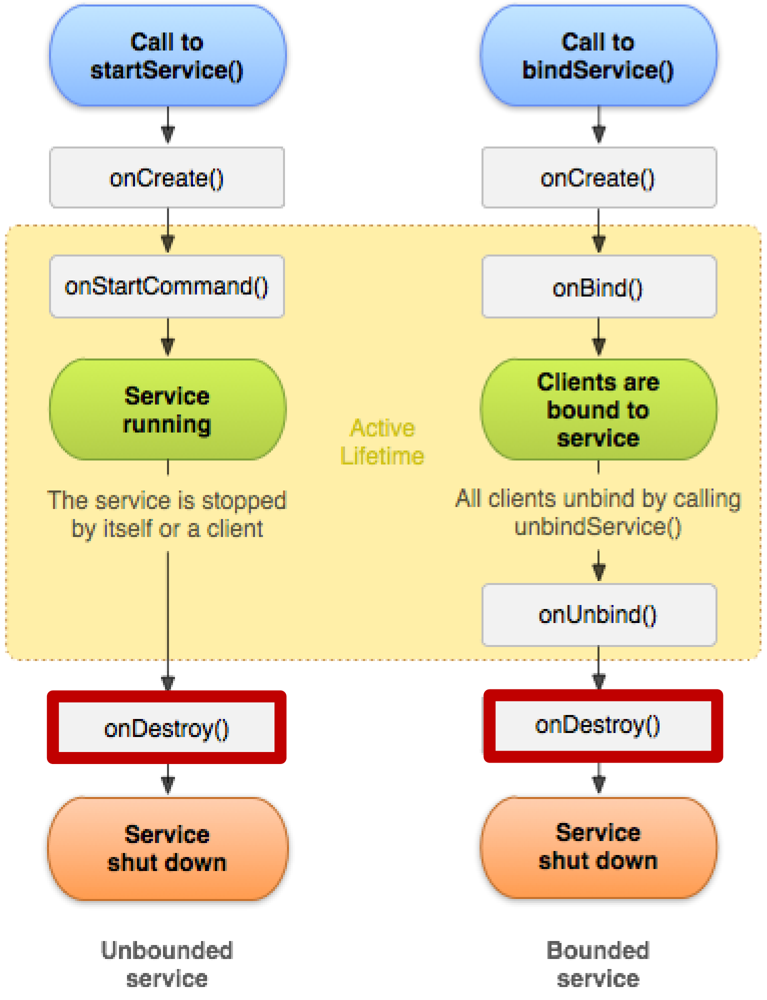
Service Lifecycle Hook Methods

- Services lifecycle methods include
  - **onCreate()** – called when Service process is created, by any means
  - **onStartCommand()** – called each time Service is sent a command via startService()
  - **onBind() / onUnbind** – called when a client binds/unbinds to Service via bindService()/unBindService()



Service Lifecycle Hook Methods

- Services lifecycle methods include
  - **onCreate()** – called when Service process is created, by any means
  - **onStartCommand()** – called each time Service is sent a command via startService()
  - **onBind() / onUnbind** – called when a client binds/unbinds to Service via bindService()/unBindService()
  - **onDestroy()** – called as Service is being shut down to cleanup resources



Services do not automatically run in their own processes or threads

ThreadedDownloadService Example

```
public class ThreadedDownloadService extends Service {
```

Inherit from Service class

```
  public int onStartCommand(Intent intent, int flags, int startId) {
    super.onStartCommand( intent, flags, startId );
    String downloadType = intent.getCharSequenceExtra
                                ("DOWNLOAD_TYPE").toString();
      if (downloadType.equals("messenger"))
        threadMessengerDownload(intent);
      else if (downloadType.equals("pending_intent"))
        threadPendingIntentDownload( intent );
      else if (downloadType.equals("asynctask")
        asyncTaskDownload(intent);

      return Service.START_STICKY;
    }
```

Lifecycle hook method downloads image via various concurrency & IPC mechanisms

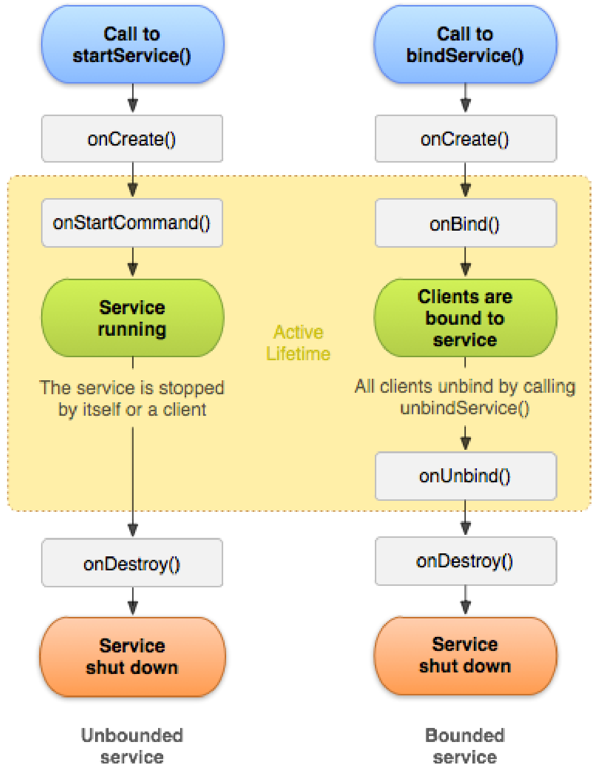
Instruct Android to run ThreadedDownloadService in its own process

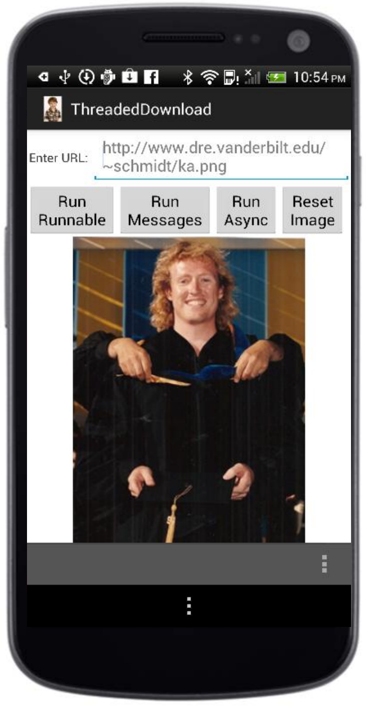
```
<service android:name="ThreadedDownloadService“
         android:process=":my_process"/>
```

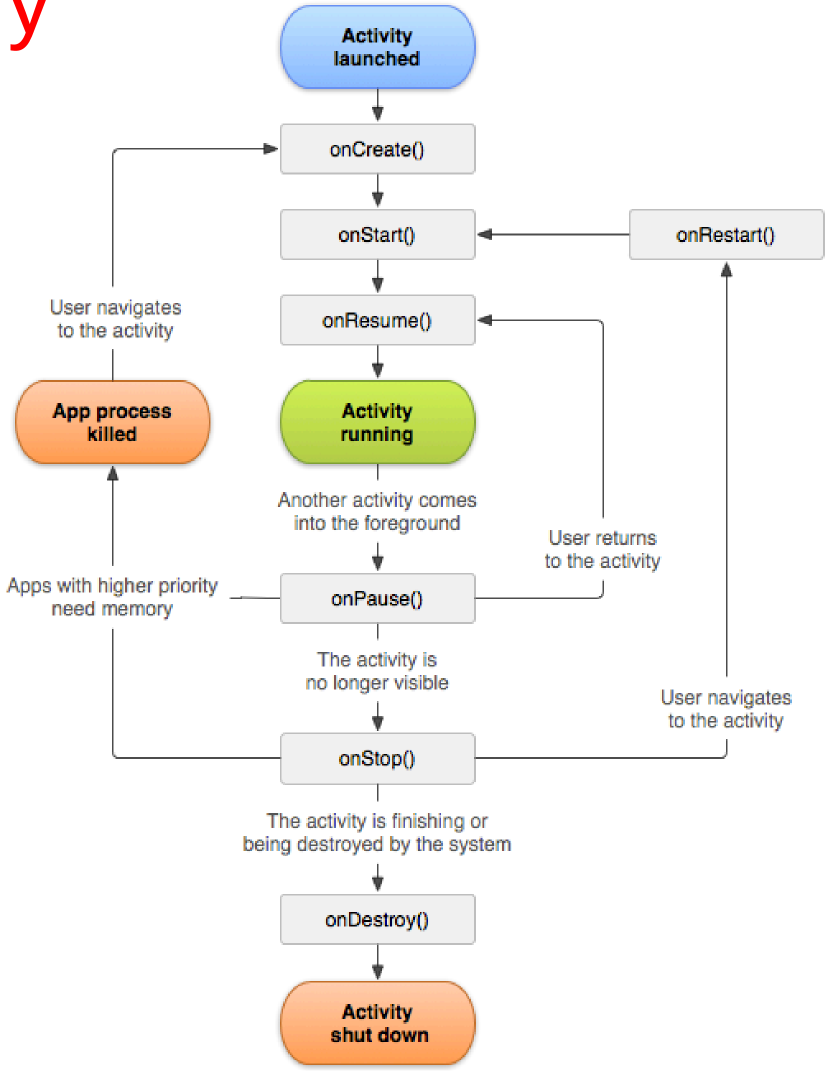
Note the “inversion of control” in the Android Service framework

Summary

- Android's framework components support inversion of control & embody many commonalities & variabilities of mobile app development

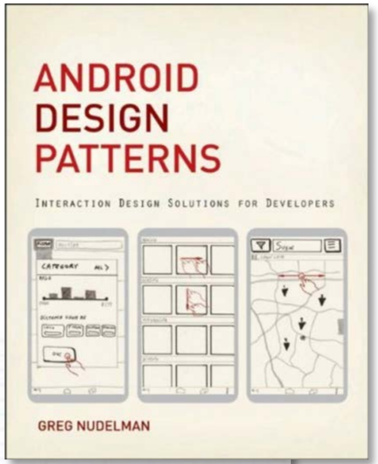





Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

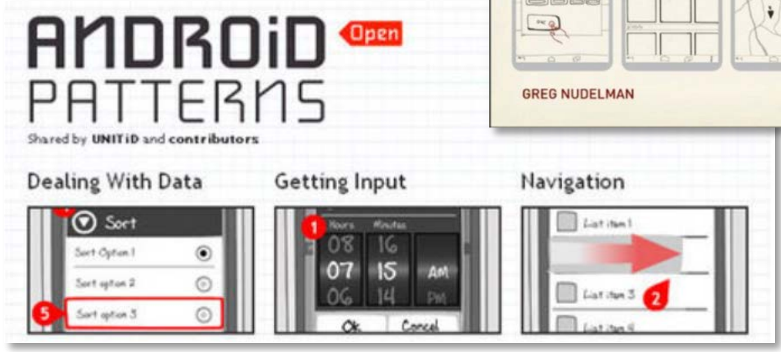
Summary

- Android's framework components support inversion of control & embody many commonalities & variabilities of mobile app development
- There are many patterns in Android
  - Both at the infrastructure & app levels


ANDROID DESIGN PATTERNS

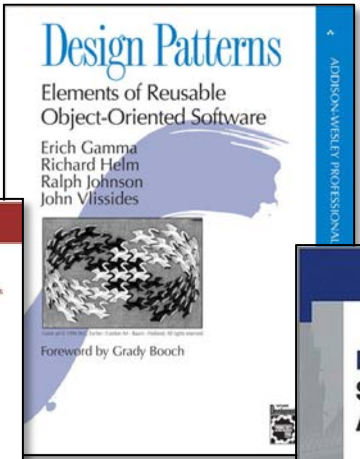
INTERACTION DESIGN SOLUTIONS FOR DEVELOPERS

GREG NUDELMAN


ANDROID Open PATTERNS

Shared by UNITiD and contributors

www.androidpatterns.com

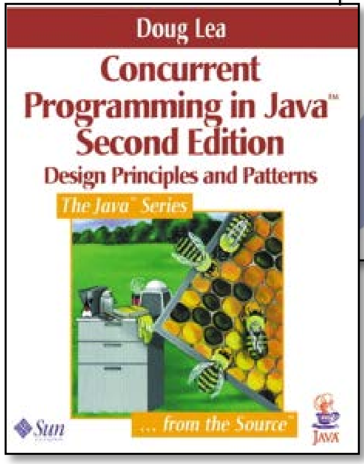

Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL

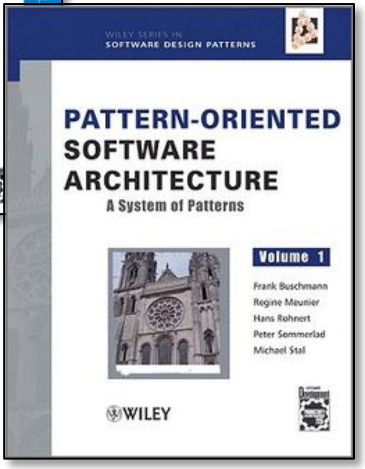

Doug Lea

Concurrent Programming in Java™ Second Edition

Design Principles and Patterns

The Java™ Series

... from the Source™


WILEY SERIES IN SOFTWARE DESIGN PATTERNS

PATTERN-ORIENTED SOFTWARE ARCHITECTURE

A System of Patterns

Volume 1

Frank Buschmann
Regine Meunier
Hans Rohnert
Peter Sommerlad
Michael Stal

WILEY

ANDROID DESIGN PATTERNS

www.androiddesignpatterns.com