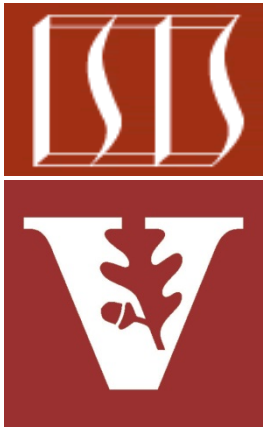
Android Services & Local IPC: Programming Started Services with Intents & Messengers (Part 1)

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



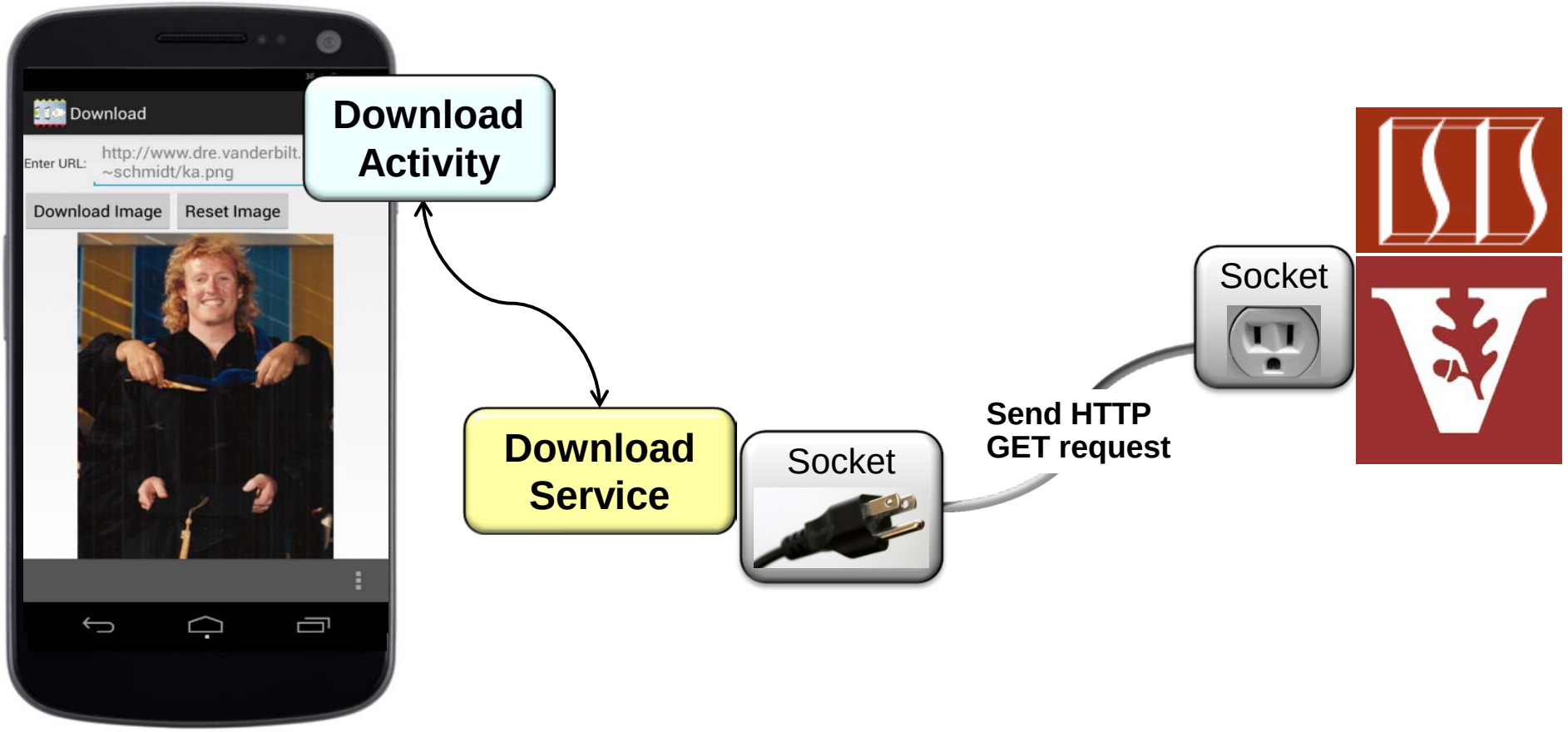
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Module

- Understand how to use a Started Service, Intent, & Messengers to design a Download Application that retrieves/displays images from a remote server

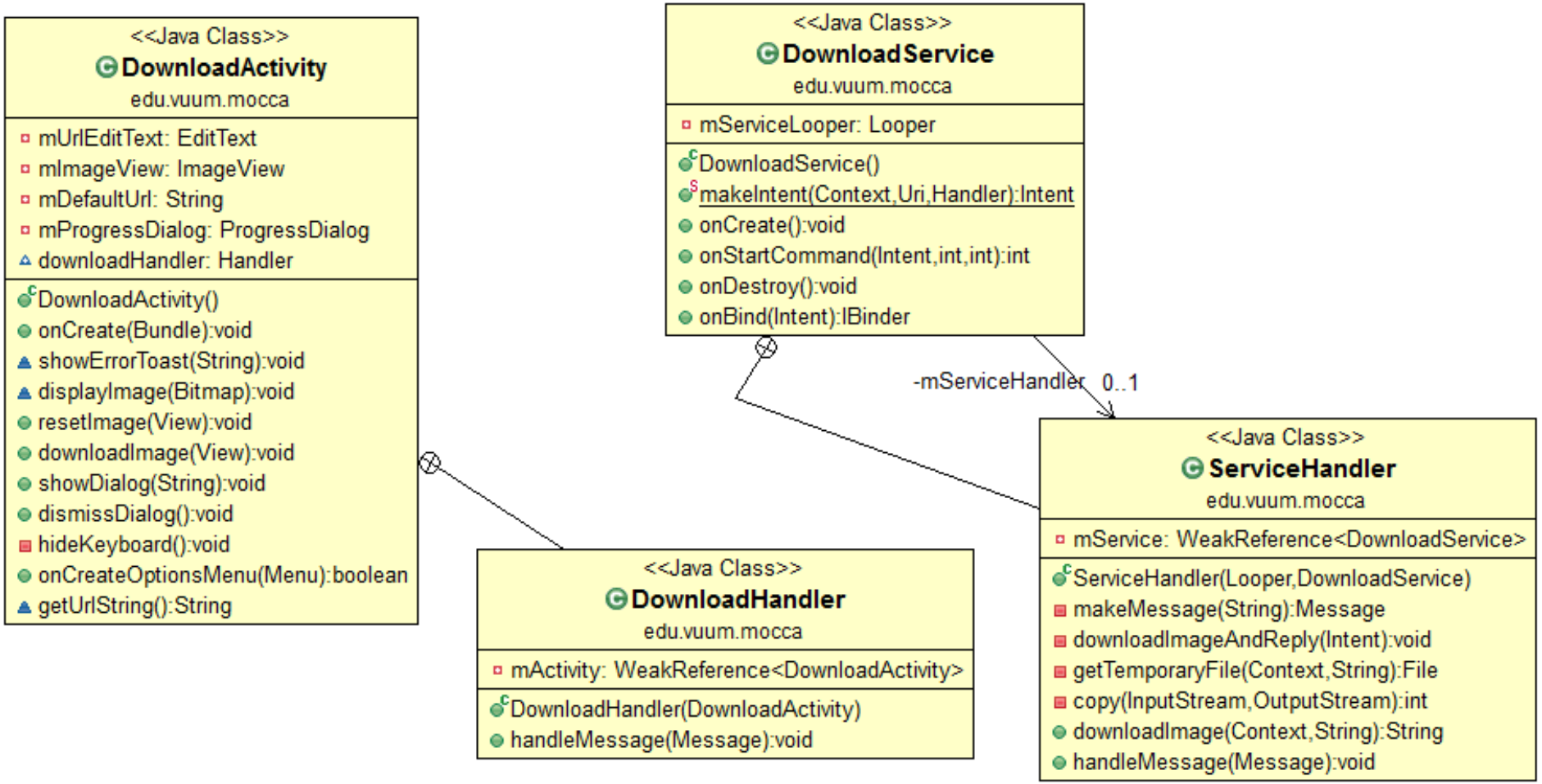


This part shows how to program the Download Application with Activities & Started Services

Overview of the Download Application

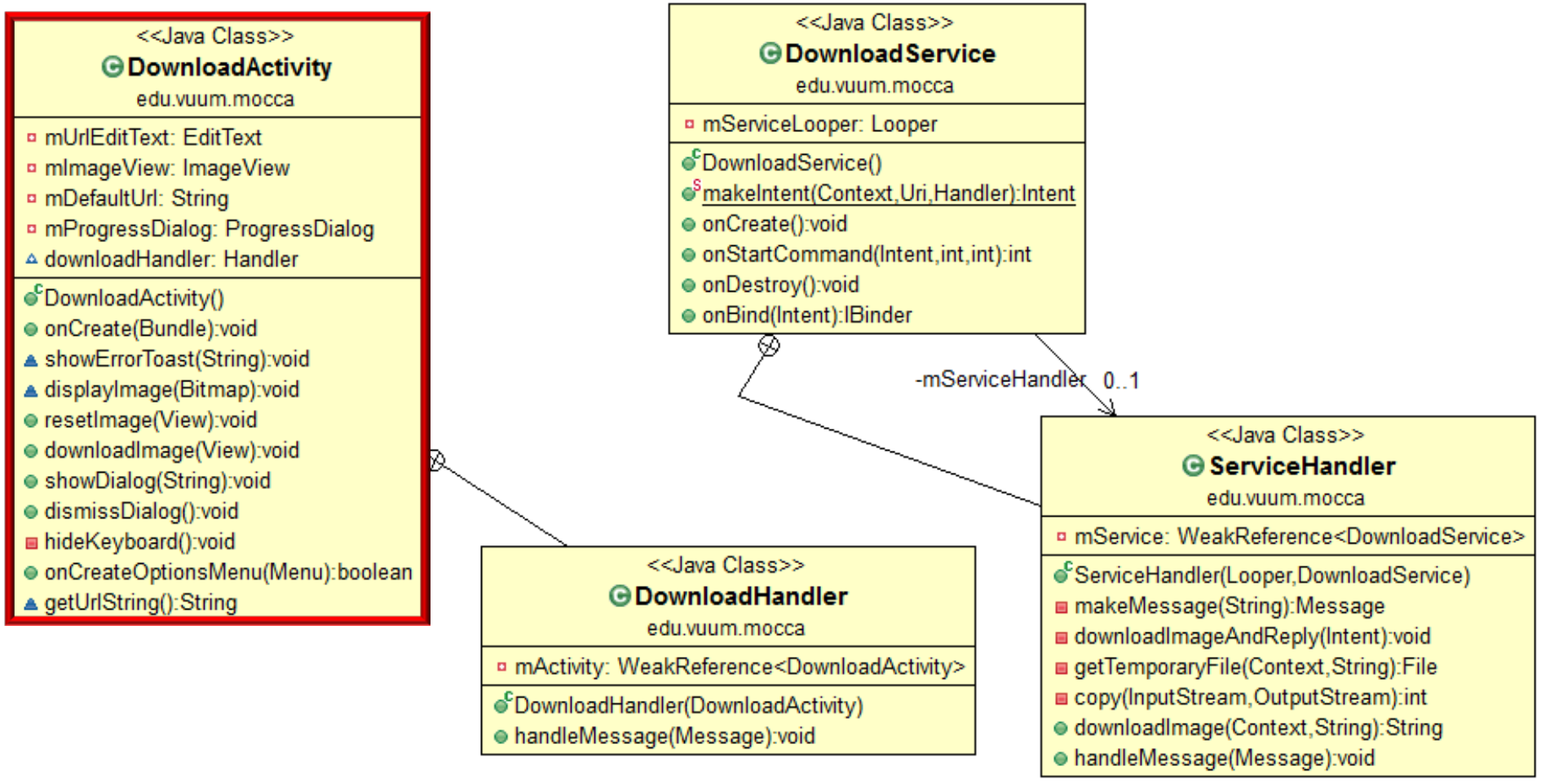
Overview of the Download Application

- Download Application uses a Started Service, Intent, & Messengers



Overview of the Download Application

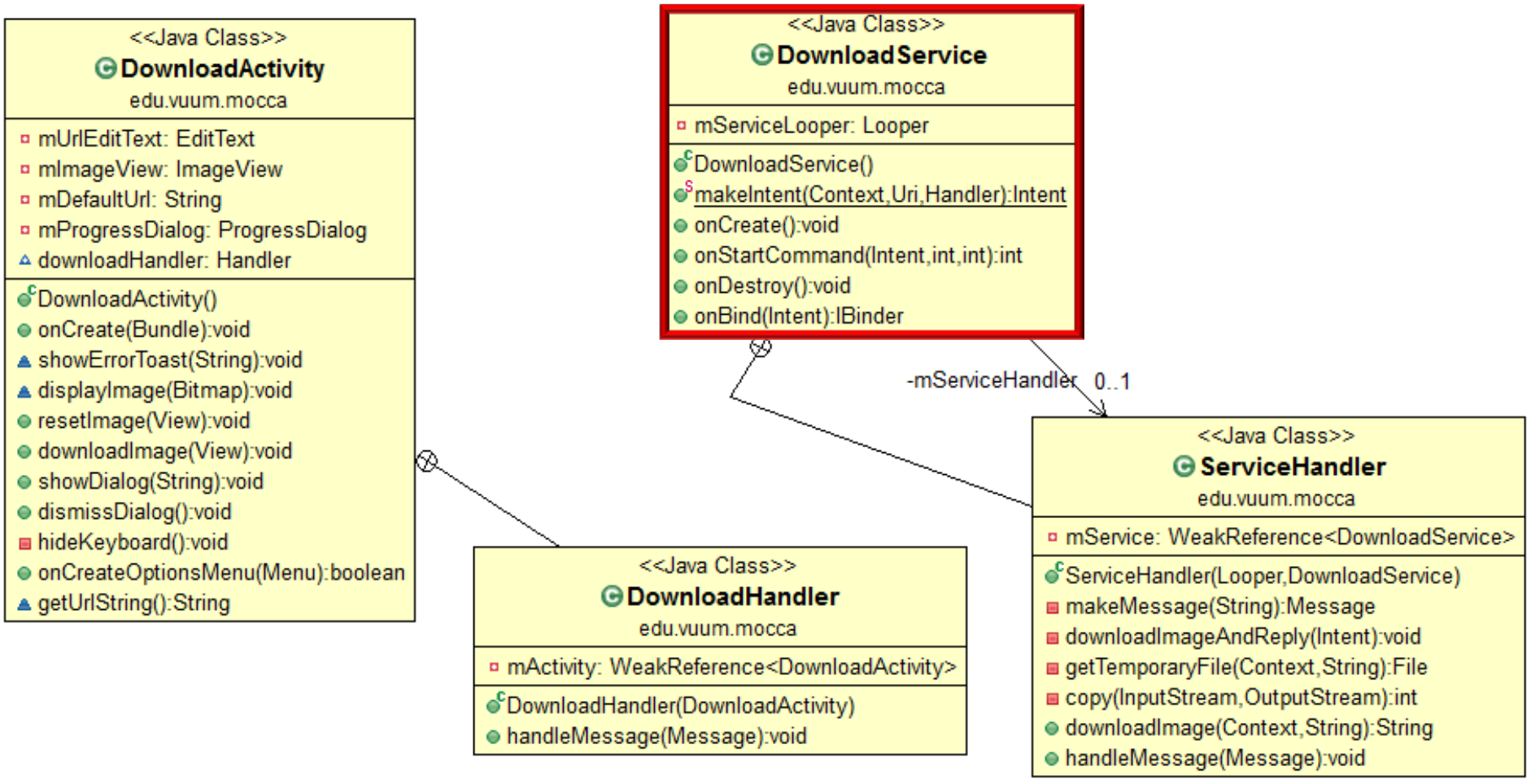
- Download Application uses a Started Service, Intent, & Messengers



Enables a user to download & display a bitmap image via the DownloadService

Overview of the Download Application

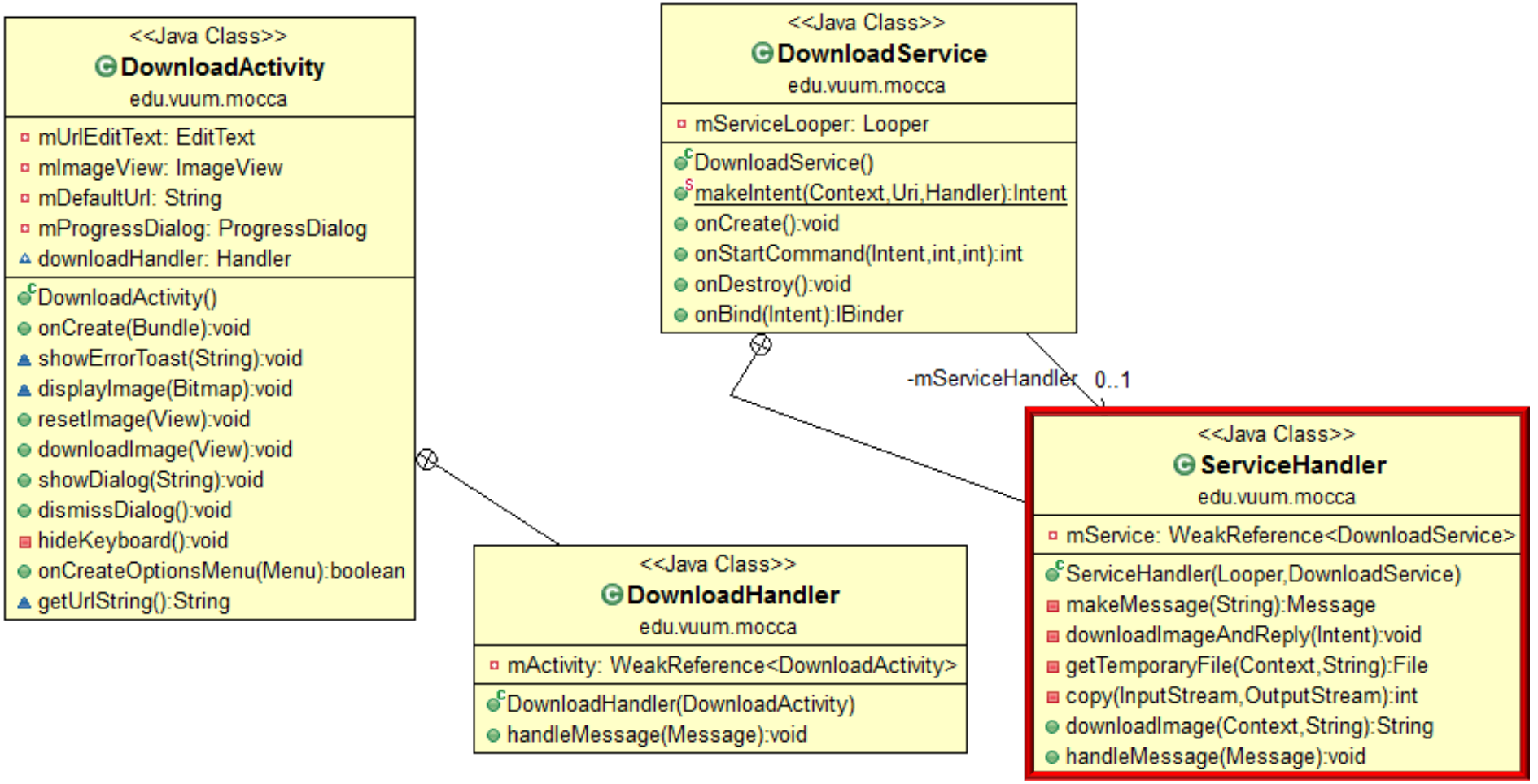
- Download Application uses a Started Service, Intent, & Messengers



Downloads & stores a bitmap image on behalf of the DownloadActivity

Overview of the Download Application

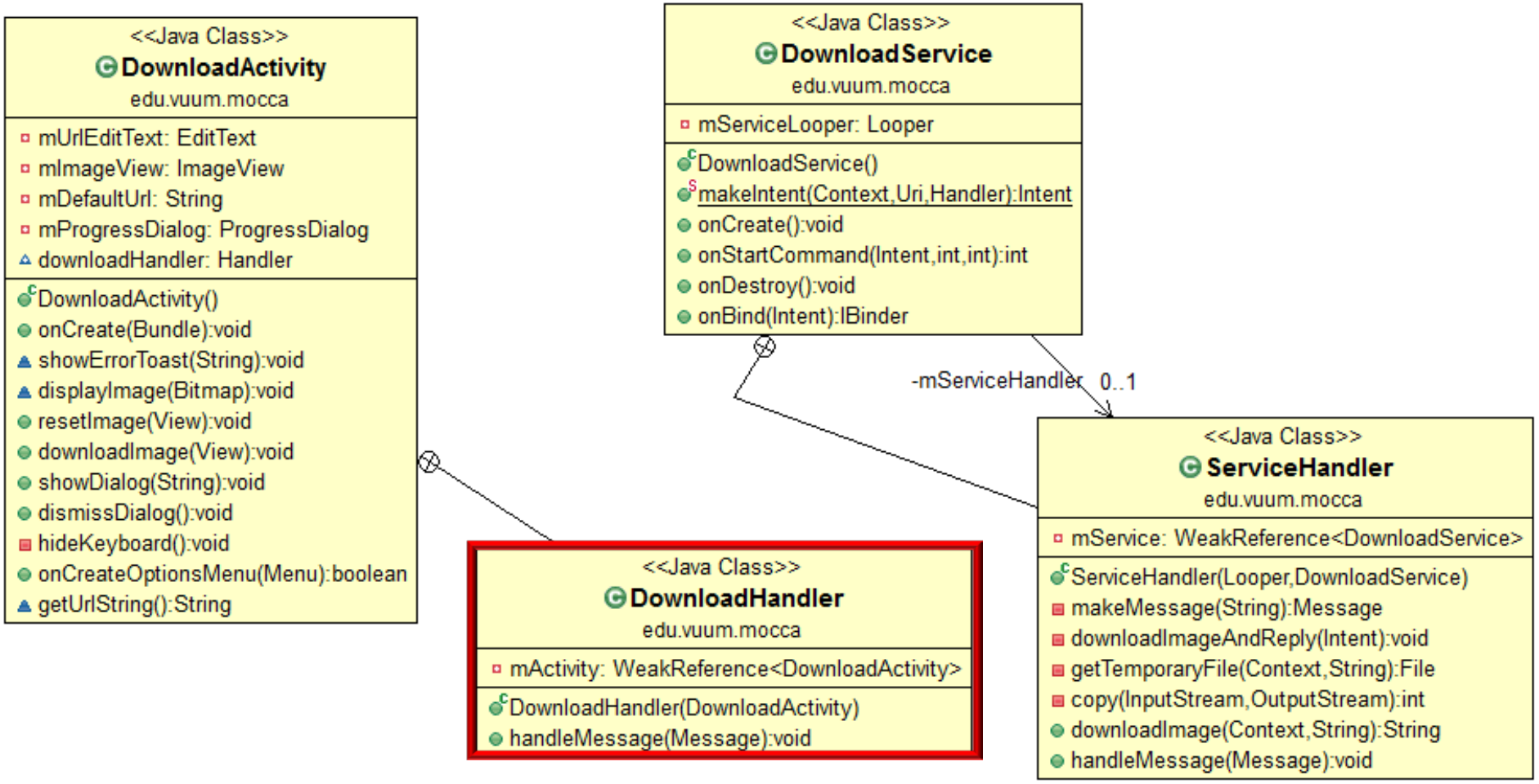
- Download Application uses a Started Service, Intent, & Messengers



Processes messages sent via onStartCommand() that indicate which images to download, store, & return

Overview of the Download Application

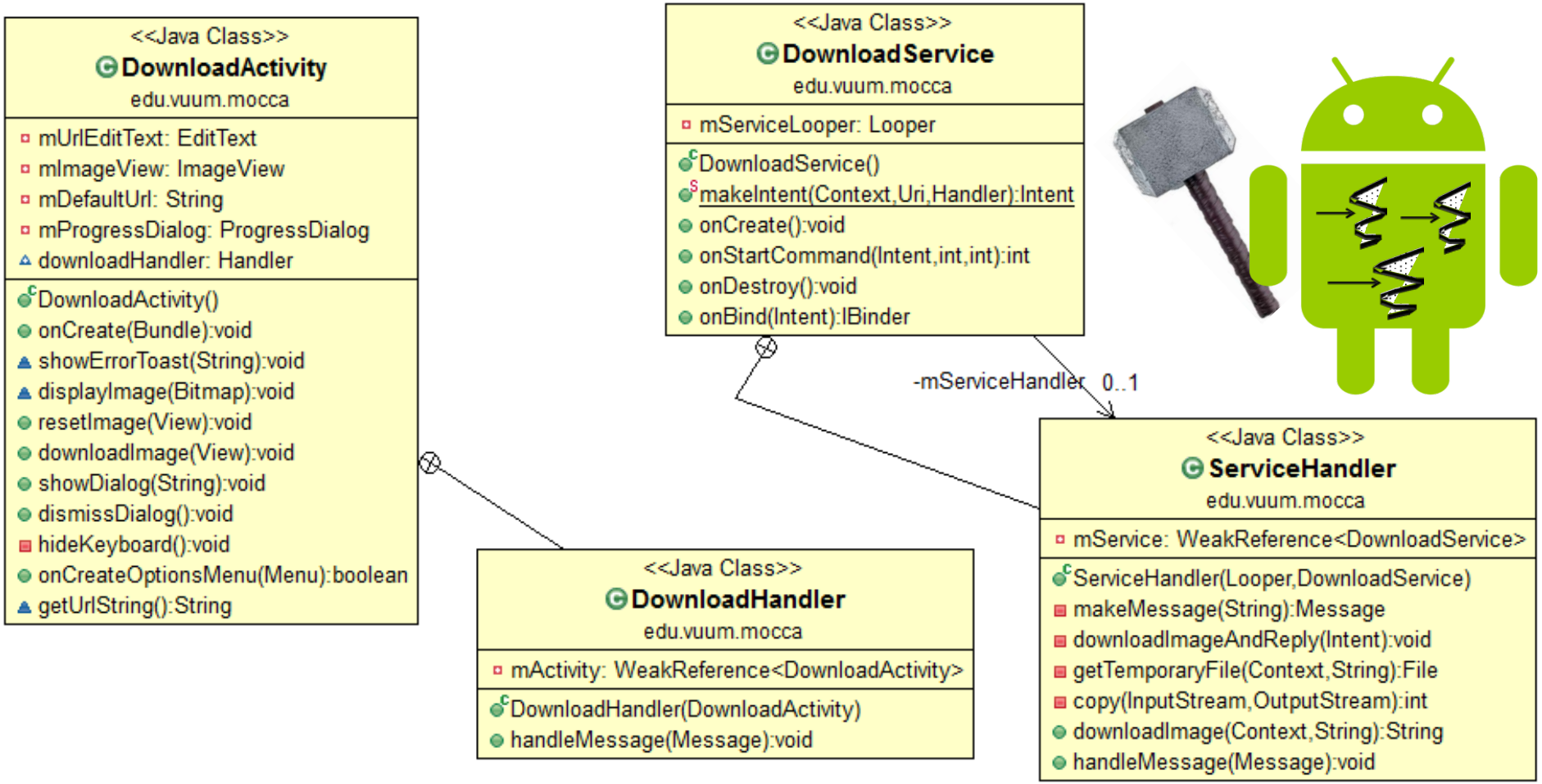
- Download Application uses a Started Service, Intent, & Messengers



A nested class that inherits from Handler & displays images retrieves & stored by the DownloadService

Overview of the Download Application

- Download Application uses a Started Service, Intent, & Messengers
  - This example is complex & uses many classes & Android mechanisms



Overview of the Download Application

- Download Application uses a Started Service, Intent, & Messengers
  - This example is complex & uses many classes & Android mechanisms
  - We therefore analyze it from various perspectives

Overview of the Download Application

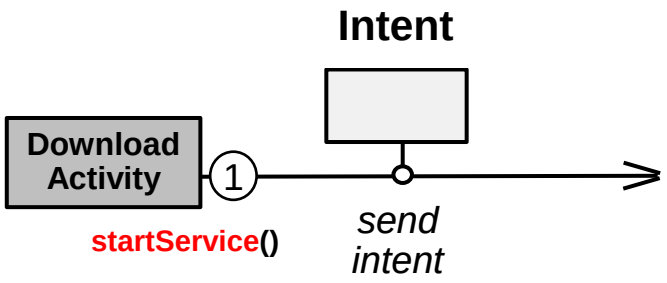
- Download Application uses a Started Service, Intent, & Messengers
  - This example is complex & uses many classes & Android mechanisms
  - We therefore analyze it from various perspectives
  - Run/read the code & watch the video carefully to understand how it works

See github.com/douglascraigschmidt/POSA-14/tree/master/ex/DownloadApplication

Design of the Download Application

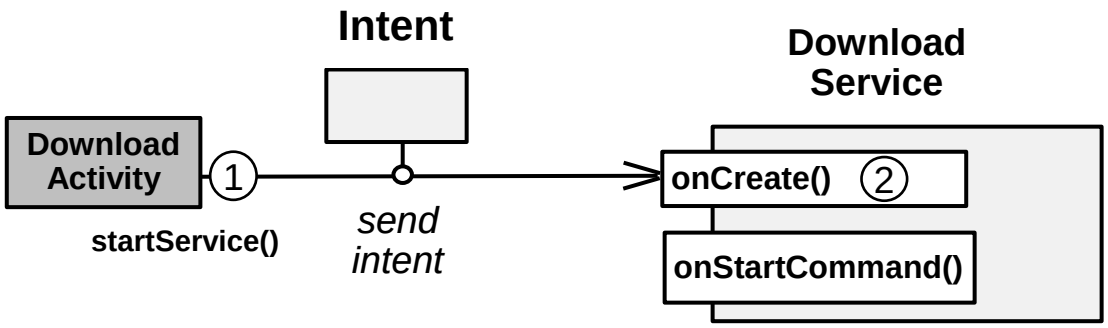
Design of the Download Application

- DownloadActivity sends an Intent via startService()



Design of the Download Application

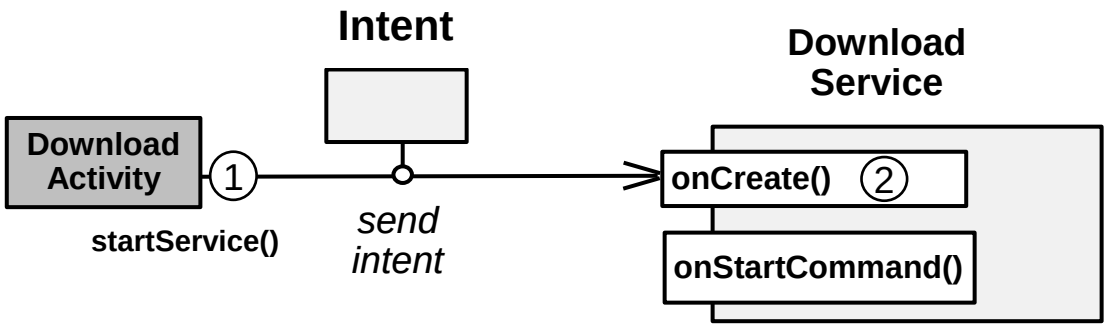
- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
  - Based on the *Activator* pattern



See www.dre.vanderbilt.edu/~schmidt/PDF/Activator.pdf

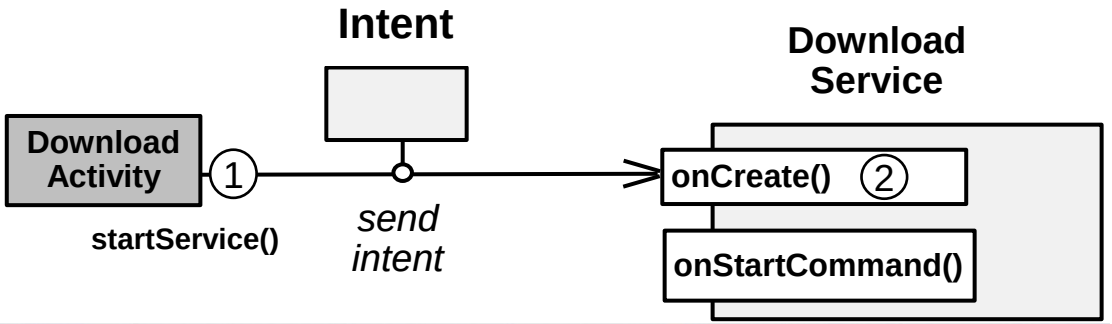
Design of the Download Application

- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
  - Based on the *Activator* pattern
    - Efficiently & transparently automate scalable on-demand activation & deactivation of services accessed by clients



Design of the Download Application

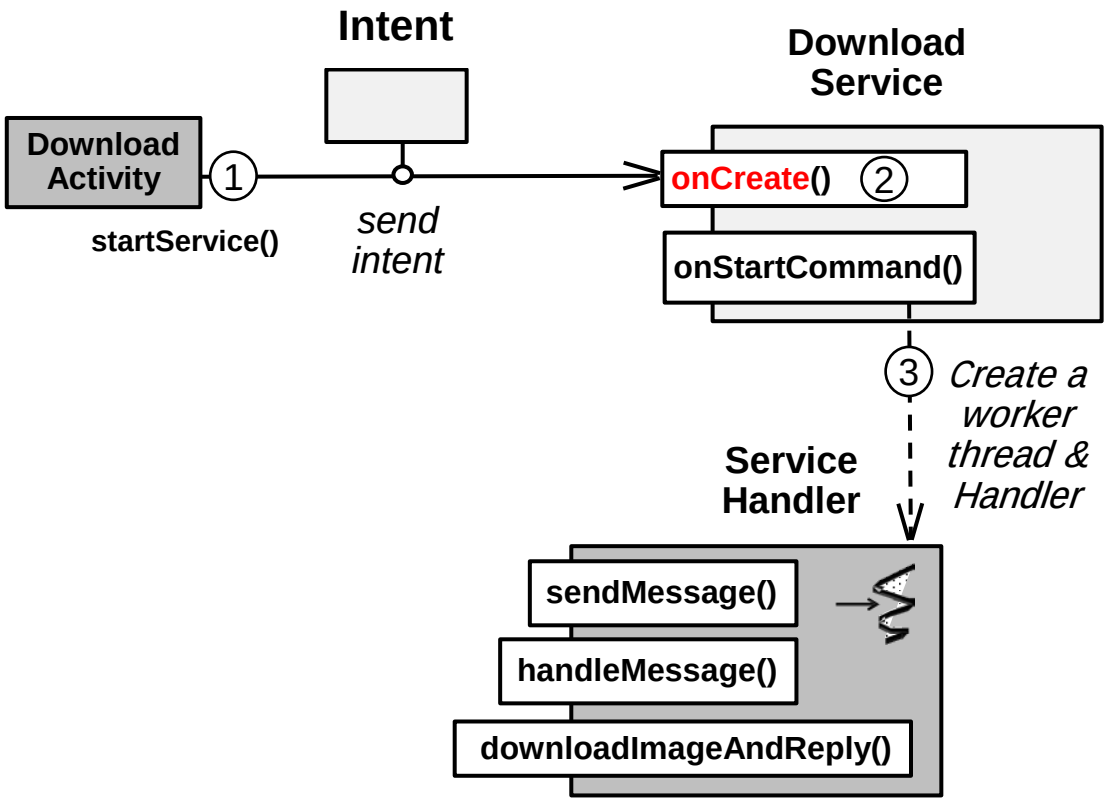
- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions



Design of the Download Application

- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions

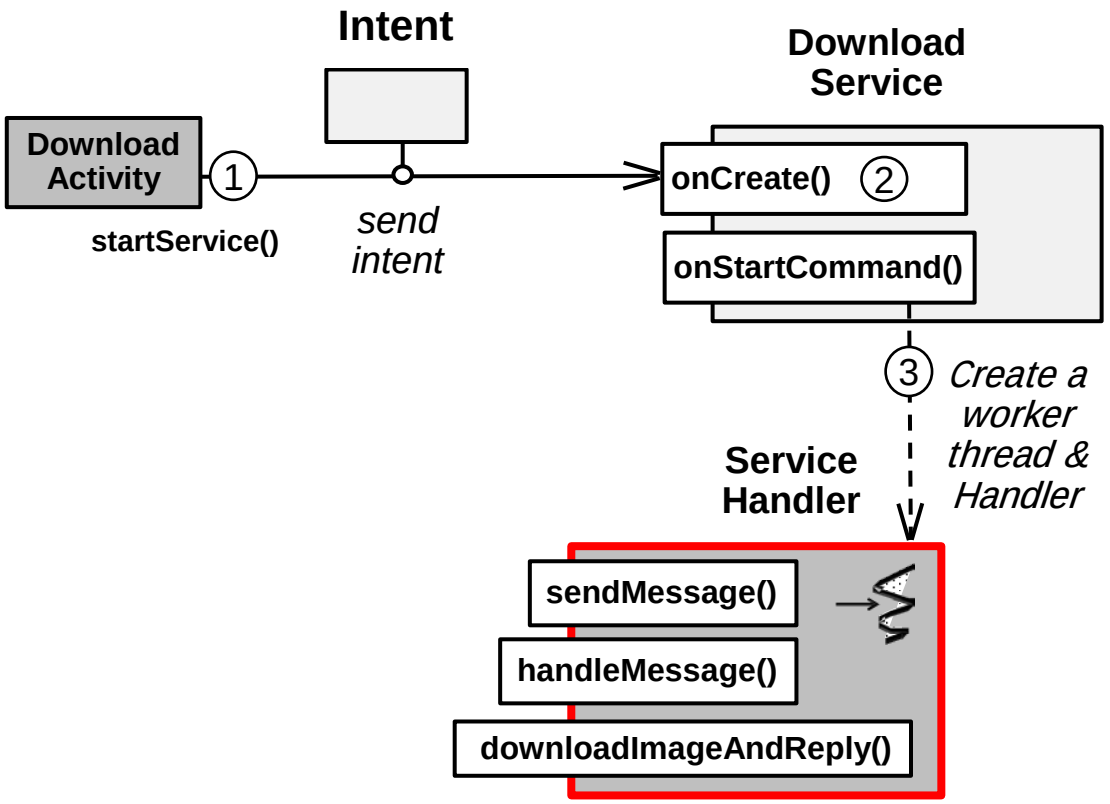
  1. Creates a ServiceHandler
     - Associated with a single HandlerThread



Design of the Download Application

- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions

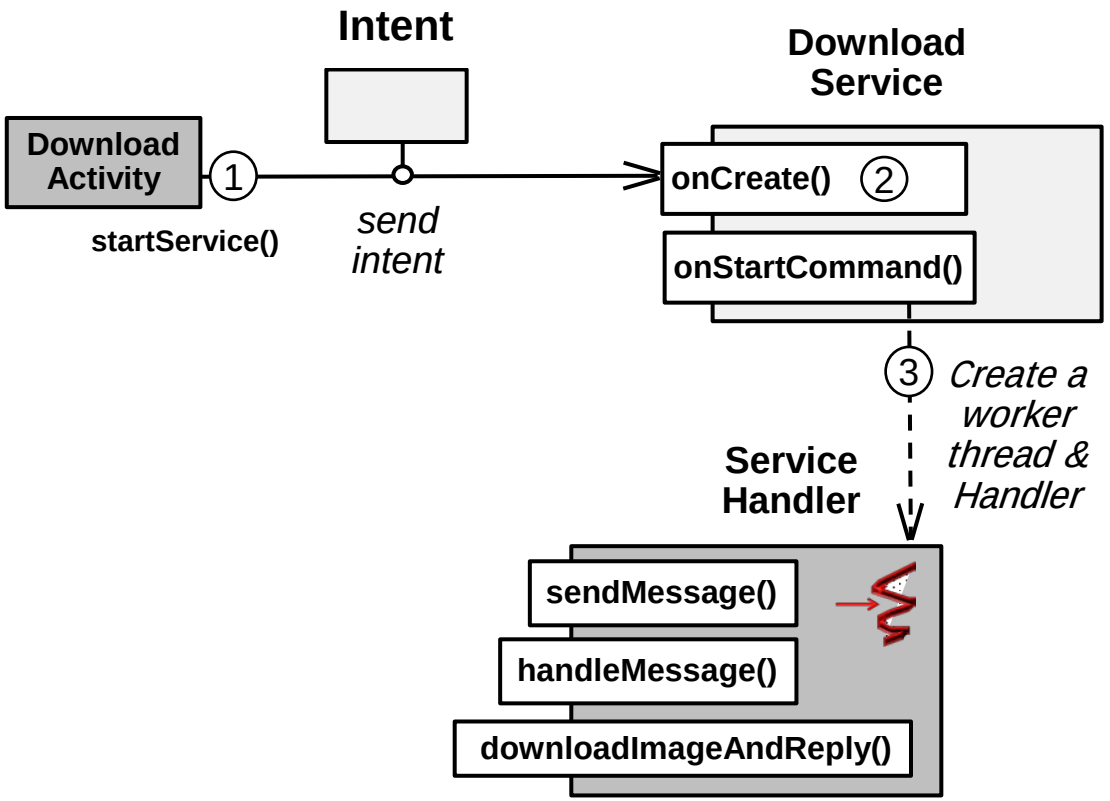
  1. Creates a ServiceHandler
     - Associated with a single HandlerThread



Design of the Download Application

- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions

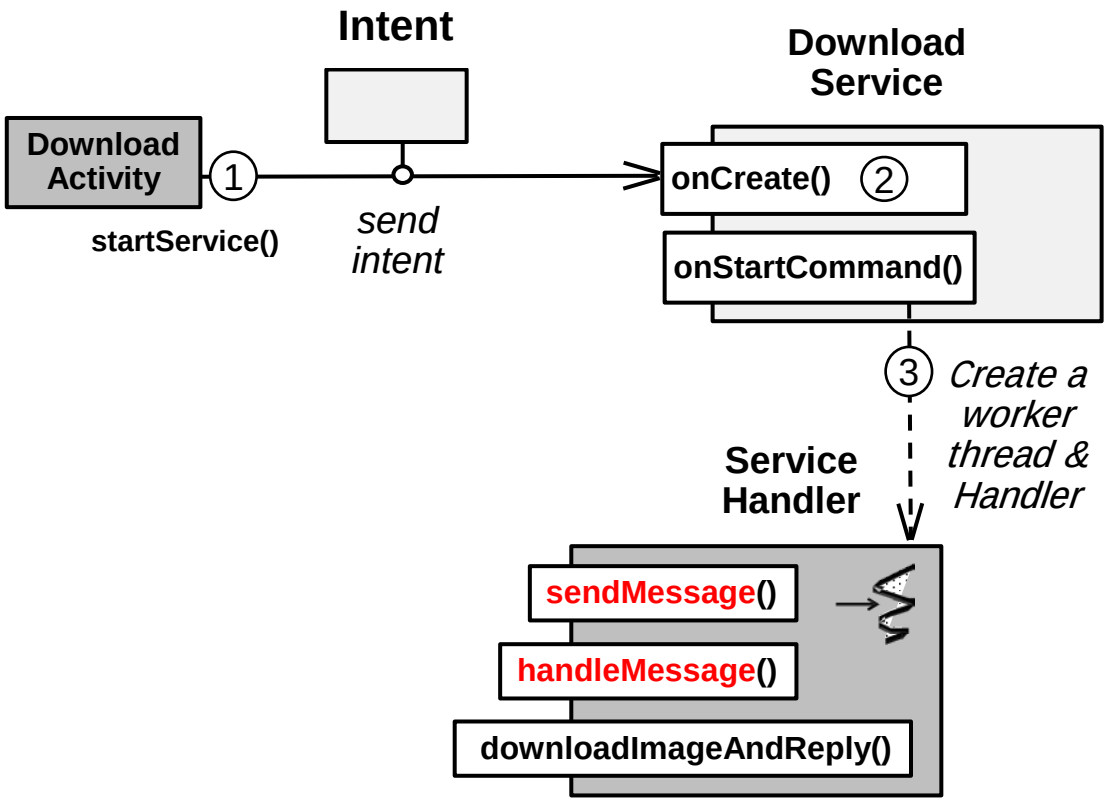
  1. Creates a ServiceHandler
     - Associated with a single HandlerThread



Design of the Download Application

- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions

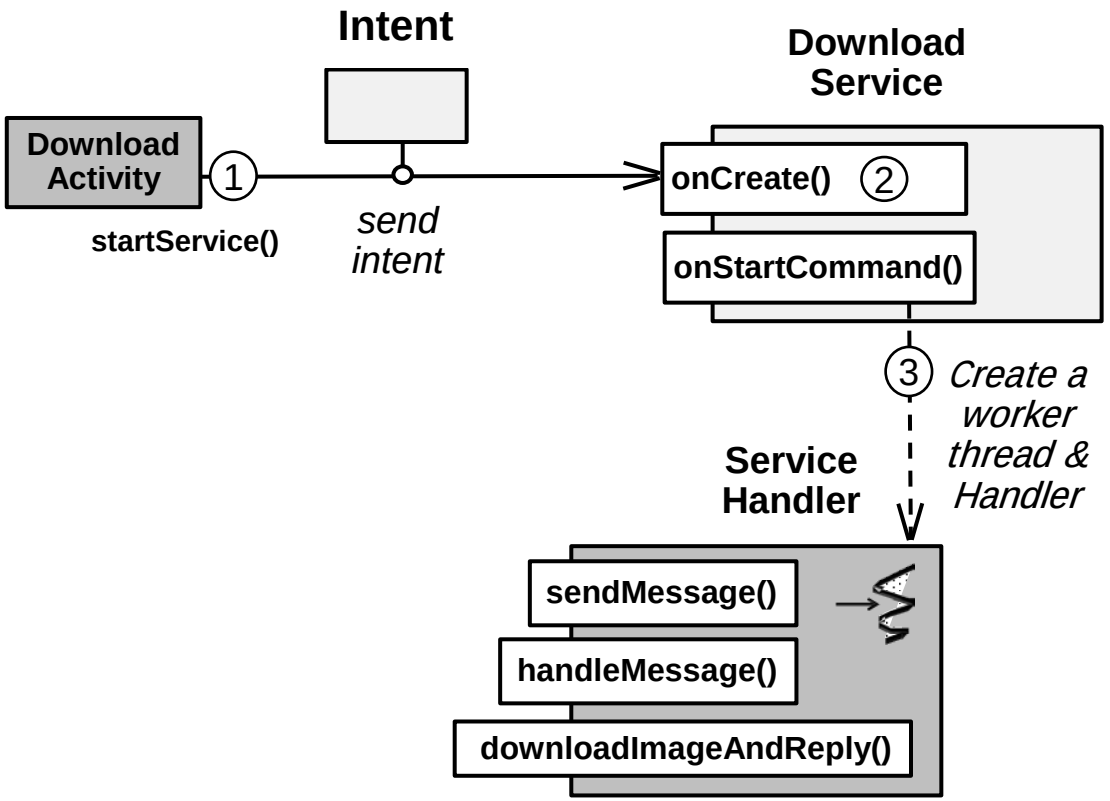
  1. Creates a ServiceHandler
     - Associated with a single HandlerThread



Design of the Download Application

- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions

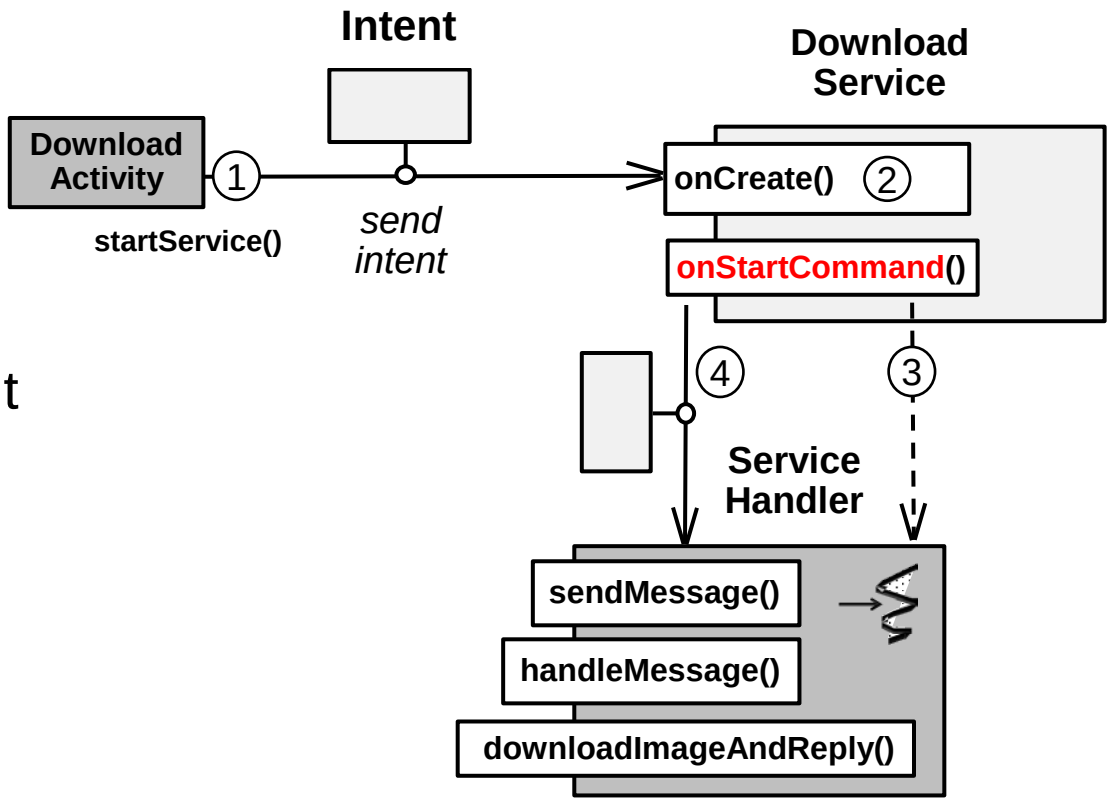
  1. Creates a ServiceHandler
     - Associated with a single HandlerThread



See earlier part on “Sending & Handling Messages with Android Handler”

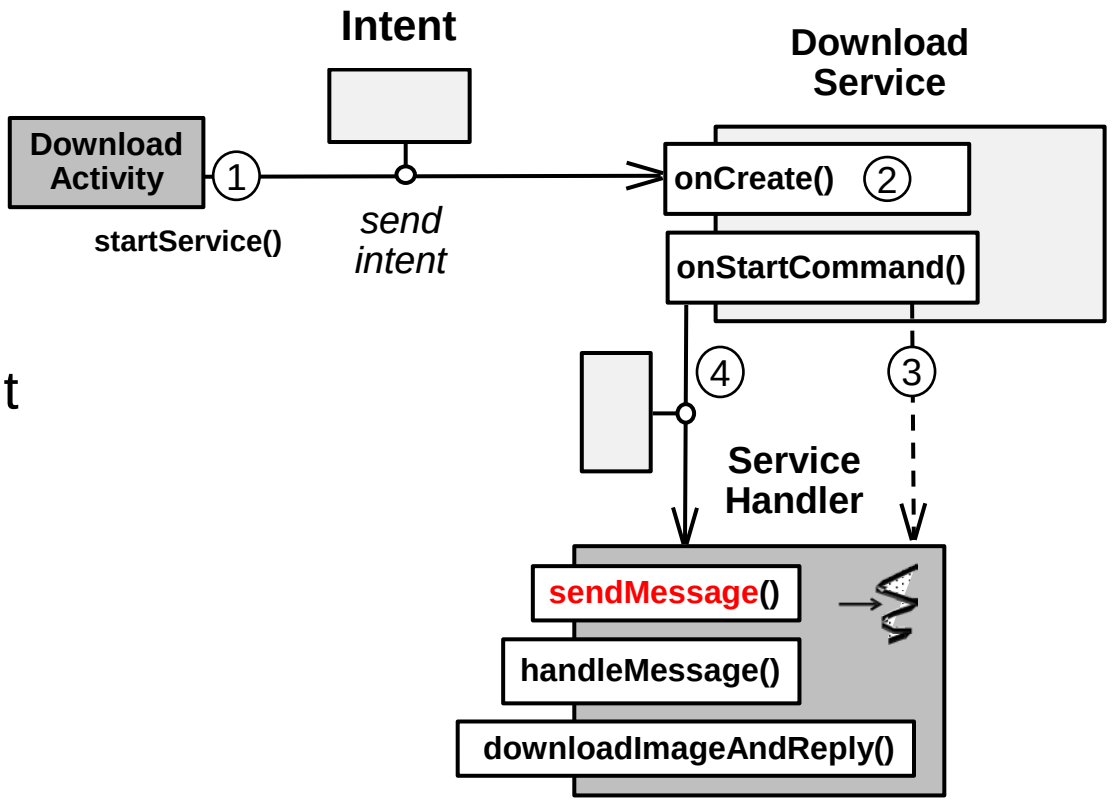
Design of the Download Application

- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
  1. Creates a ServiceHandler
  2. Receives & queues an Intent in the ServiceHandler



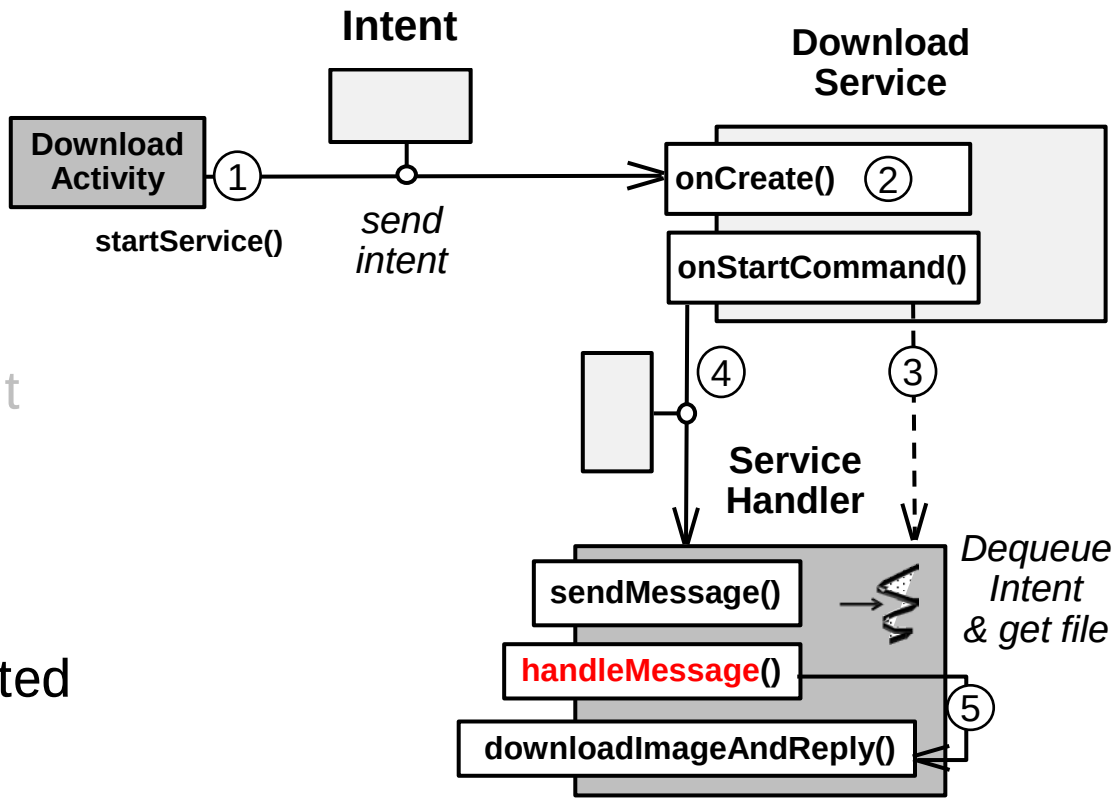
Design of the Download Application

- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
  1. Creates a ServiceHandler
  2. Receives & queues an Intent in the ServiceHandler



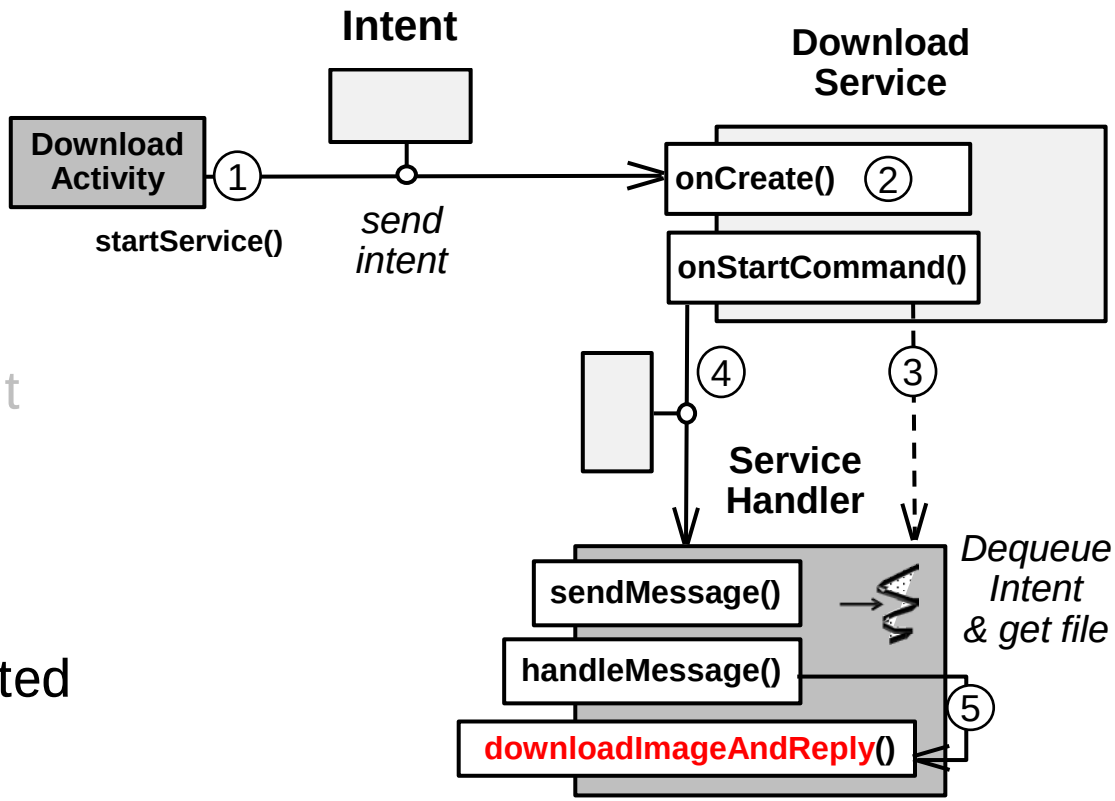
Design of the Download Application

- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
  1. Creates a ServiceHandler
  2. Receives & queues an Intent in the ServiceHandler
  3. ServiceHandler processes Intent "in the background"
     - Downloads image designated by the URL in the Intent



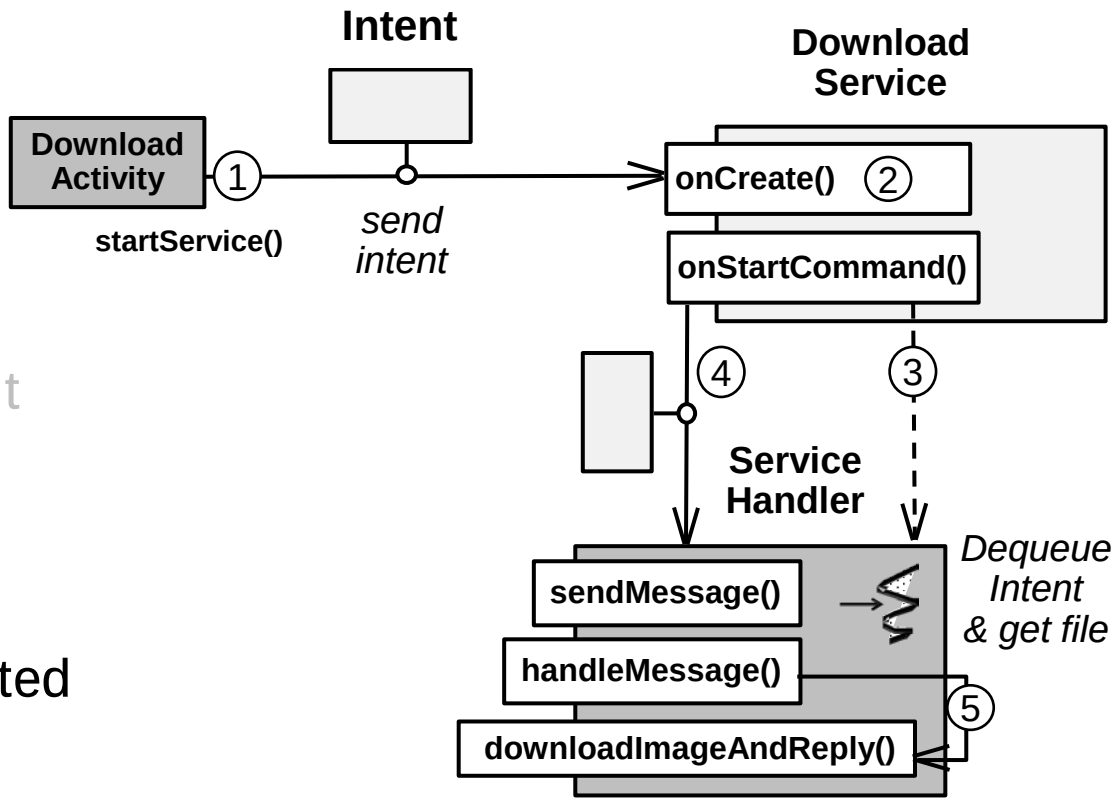
Design of the Download Application

- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
  1. Creates a ServiceHandler
  2. Receives & queues an Intent in the ServiceHandler
  3. ServiceHandler processes Intent "in the background"
     - Downloads image designated by the URL in the Intent



Design of the Download Application

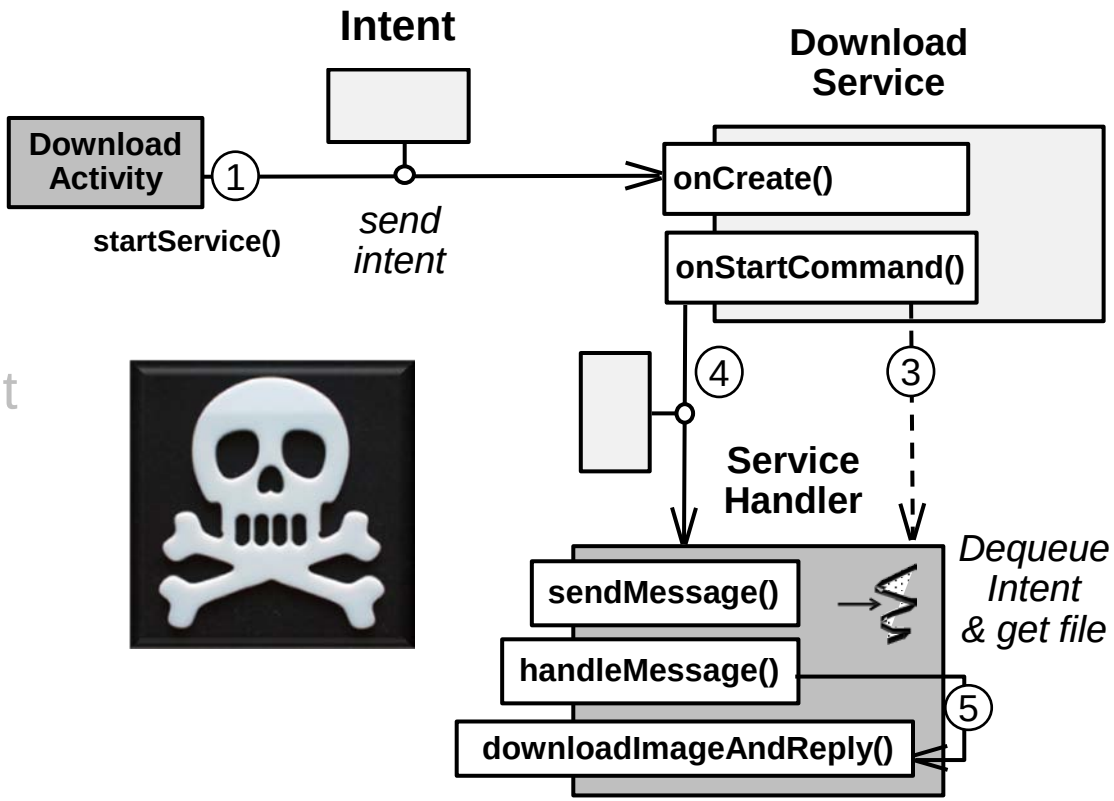
- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
  1. Creates a ServiceHandler
  2. Receives & queues an Intent in the ServiceHandler
  3. ServiceHandler processes Intent "in the background"
     - Downloads image designated by the URL in the Intent



See earlier parts on "Activity & Service Communication"

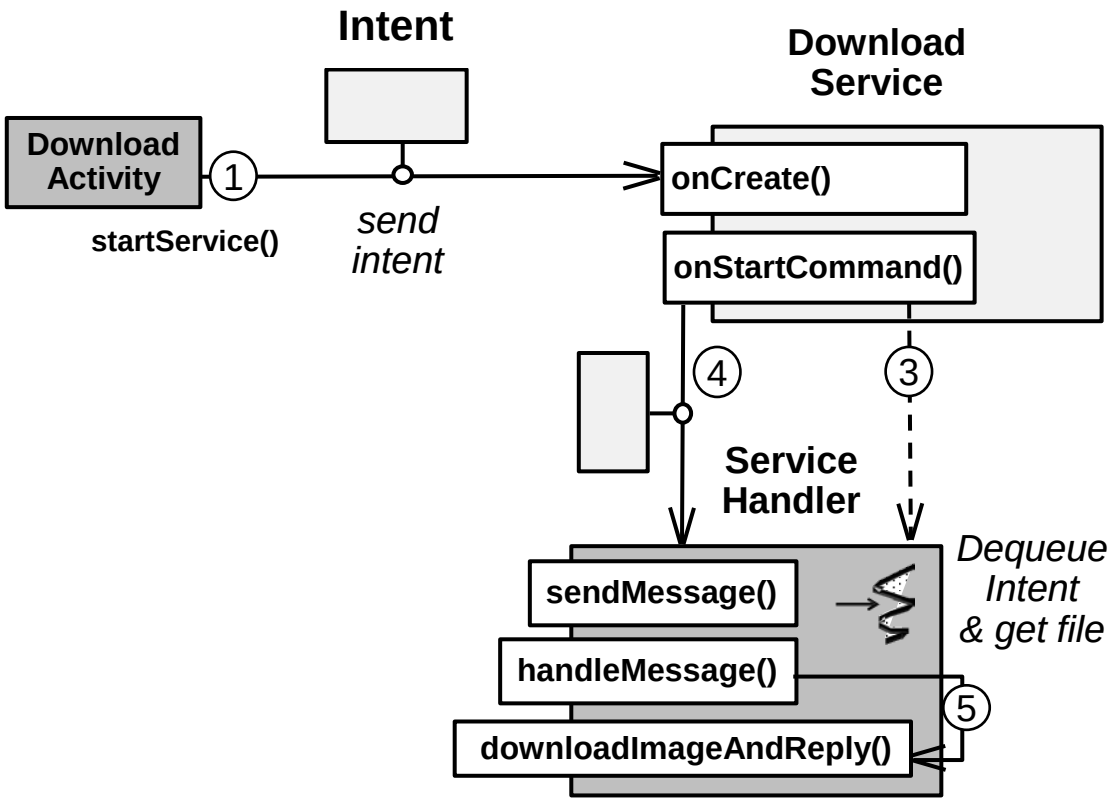
Design of the Download Application

- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
  1. Creates a ServiceHandler
  2. Receives & queues an Intent in the ServiceHandler
  3. ServiceHandler processes Intent "in the background"
  4. Stops itself when there are no more Intents to handle



Design of the Download Application

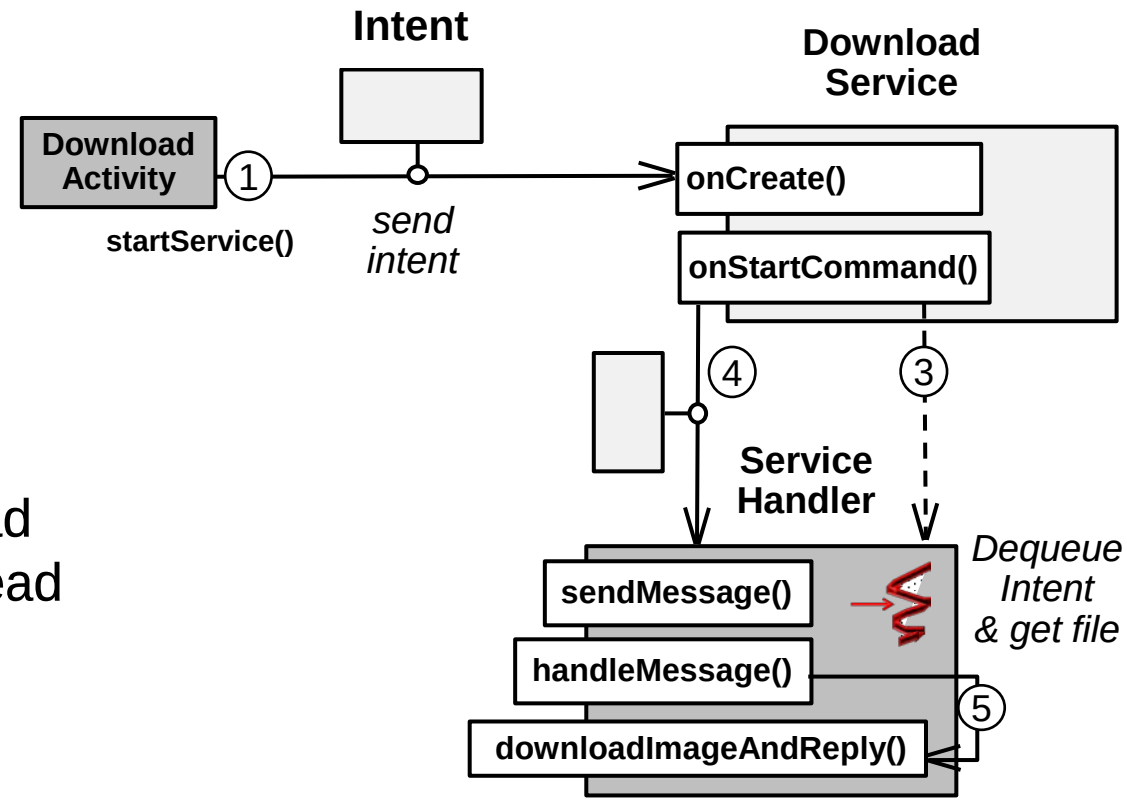
- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
- The design is guided by an Android idiom for concurrent Service processing



See en.wikipedia.org/wiki/Programming_idiom

Design of the Download Application

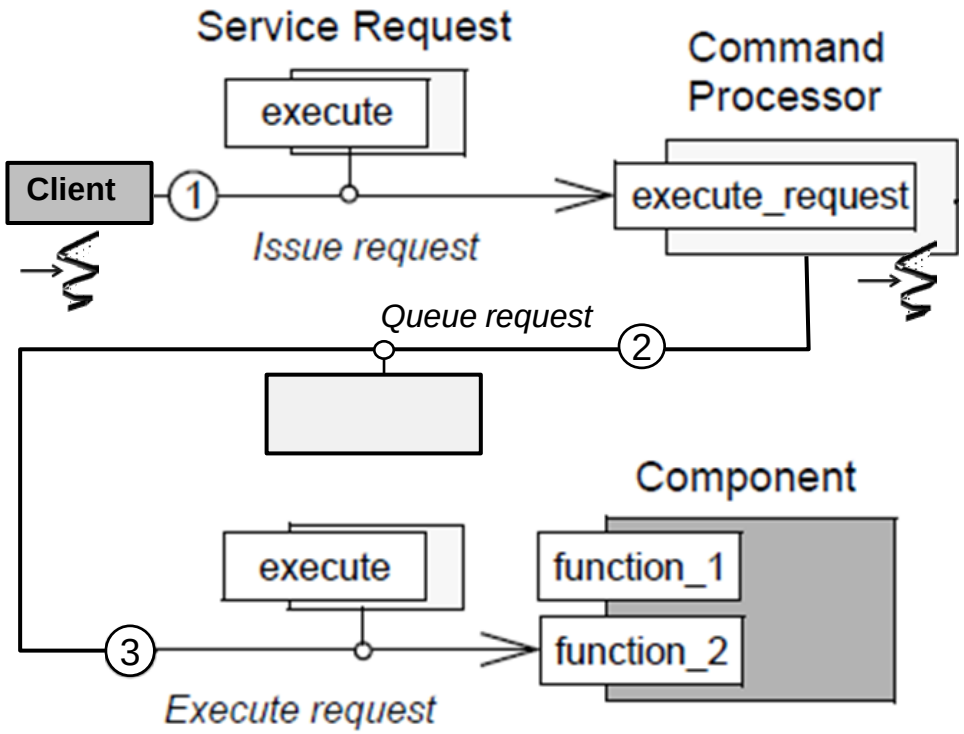
- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
- The design is guided by an Android idiom for concurrent Service processing
  - Offload tasks from UI Thread to a single background Thread



This idiom works if a Service needn't handle *multiple* requests concurrently

Design of the Download Application

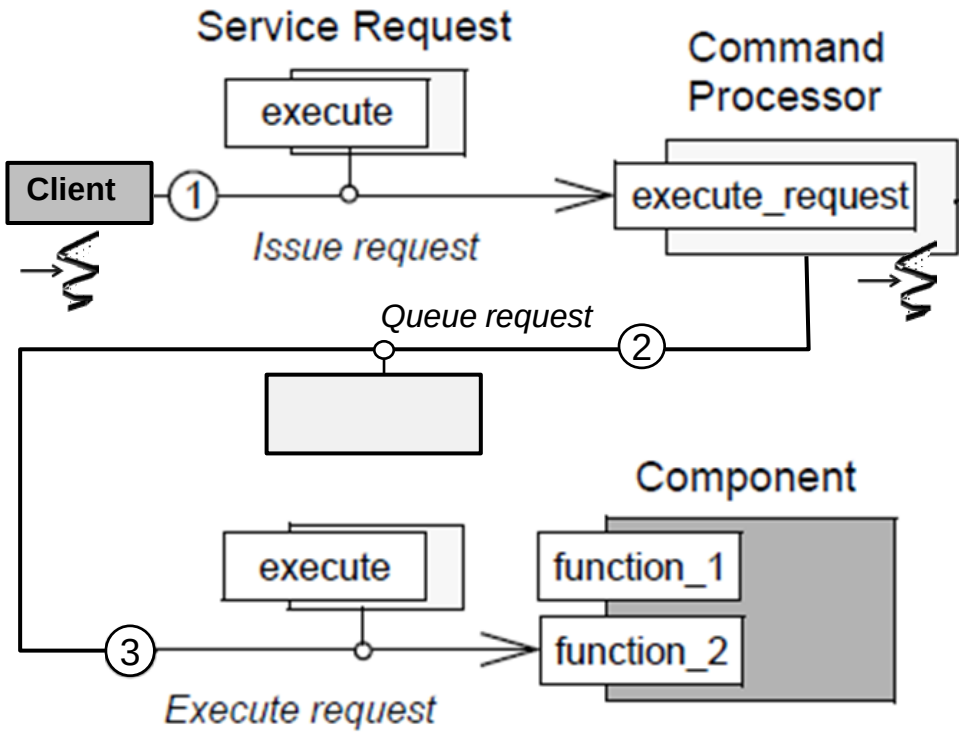
- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
- The design is guided by an Android idiom for concurrent Service processing
  - Offload tasks from UI Thread to a single background Thread
  - This design is guided by the *Command Processor* pattern



See www.dre.vanderbilt.edu/~schmidt/PDF/CommandRevisited.pdf

Design of the Download Application

- DownloadActivity sends an Intent via startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
- The design is guided by an Android idiom for concurrent Service processing
  - Offload tasks from UI Thread to a single background Thread
  - This design is guided by the *Command Processor* pattern



See upcoming parts on “The Command Processor Pattern”

Launching a Started Service

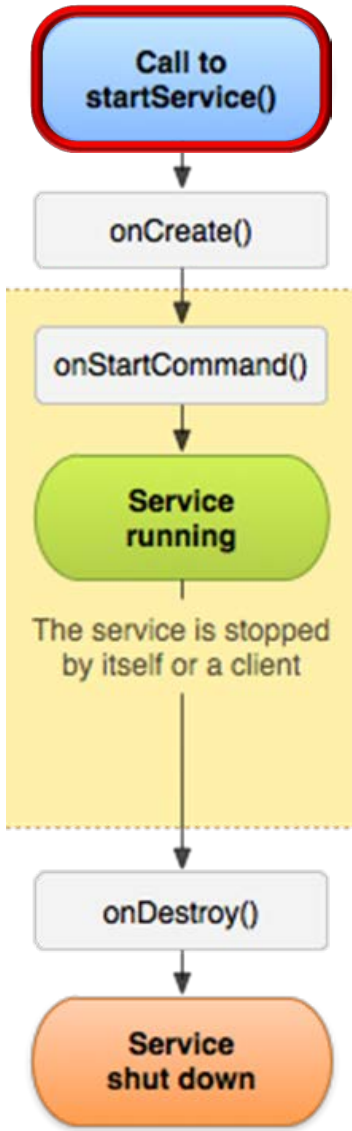
Launching a Started Service

- A client launches a Started Service by calling startService()

```
Intent intent =
  DownloadService.makeIntent
    (this, Uri.parse(url), downloadHandler);

startService(intent);
```

Download Activity



See developer.android.com/guide/components/services.html#CreatingStartedService

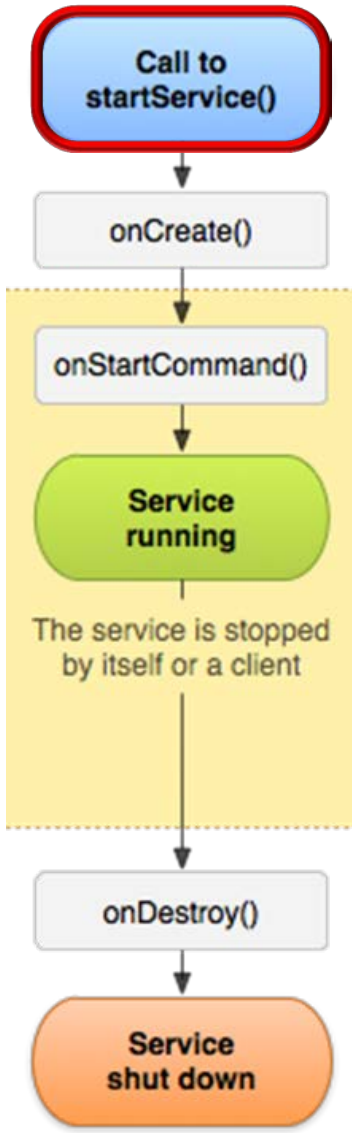
Launching a Started Service

- A client launches a Started Service by calling startService()
  - e.g., DownloadActivity creates an Intent that identifies the DownloadService & supplies a URL parameter via Intent data that tells Service what image to retrieve

```
Intent intent =
  DownloadService.makeIntent
    (this, Uri.parse(url), downloadHandler);

startService(intent);
```

Download Activity



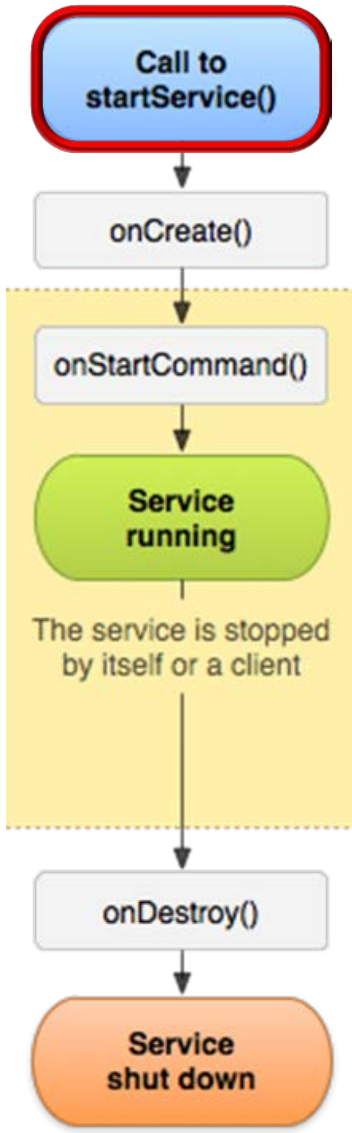
Launching a Started Service

- A client launches a Started Service by calling startService()
  - e.g., DownloadActivity creates an Intent that identifies the DownloadService & supplies a URL parameter via Intent data that tells Service what image to retrieve

```
Intent intent =
  DownloadService.makeIntent
    (this, Uri.parse(url), downloadHandler);

startService(intent);
```

Download Activity



Launching a Started Service

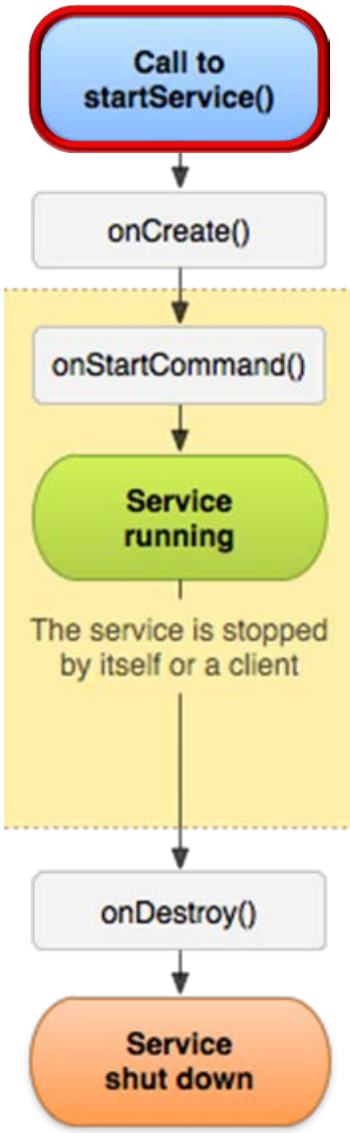
- A client launches a Started Service by calling startService()
  - e.g., DownloadActivity creates an Intent that identifies the DownloadService & supplies a URL parameter via Intent data that tells Service what image to retrieve

```
Intent intent =
  DownloadService.makeIntent
    (this, Uri.parse(url), downloadHandler);

startService(intent);
```

Download Activity

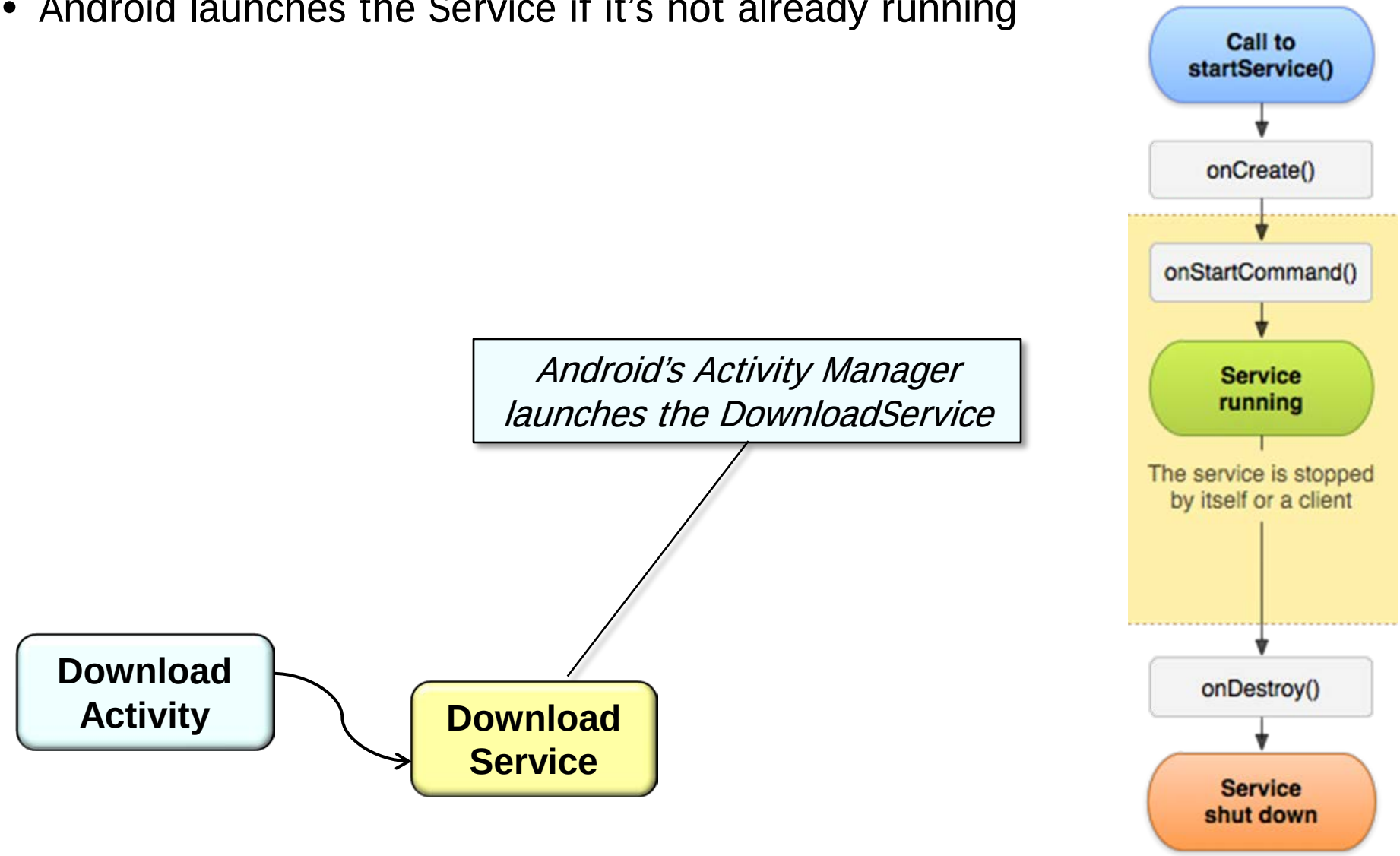
This call doesn't block the client while the DownloadService runs



Processing a Started Service (Part 1)

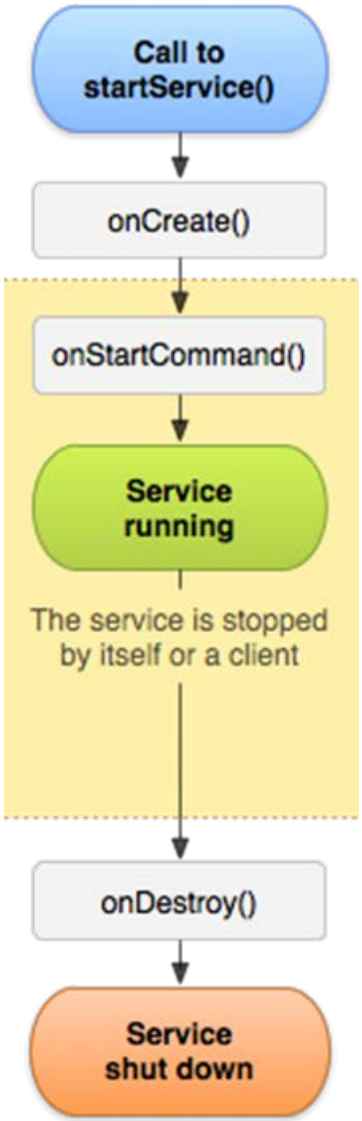
Processing a Started Service

- Android launches the Service if it's not already running



Processing a Started Service

- Android launches the Service if it's not already running
  - Applies the *Activator* pattern to efficiently & transparently automate scalable on-demand activation & deactivation of services accessed by clients



Android's Activity Manager launches the DownloadService

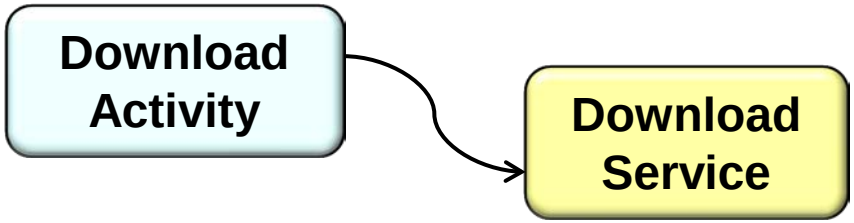
Download Activity

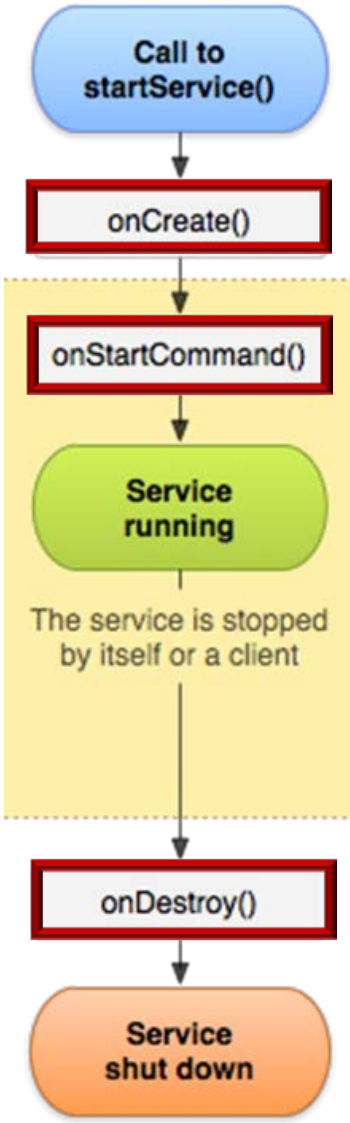
Download Service

See www.dre.vanderbilt.edu/~schmidt/PDF/Activator.pdf

Processing a Started Service

- Android launches the Service if it's not already running
- Started Services are driven by "inversion of control"

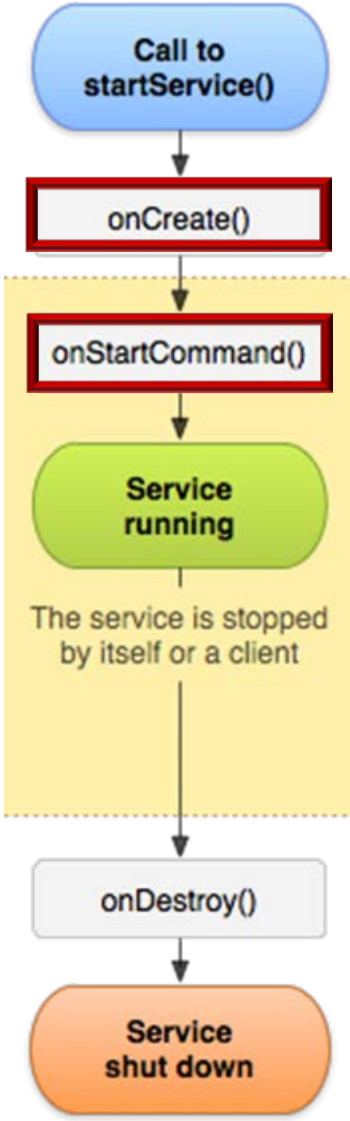




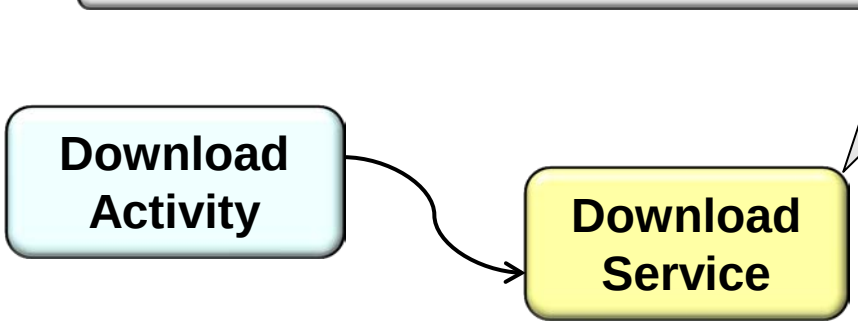
See en.wikipedia.org/wiki/Inversion_of_control

Processing a Started Service

- Android launches the Service if it's not already running
- Started Services are driven by "inversion of control"
  - e.g., the Android Service framework invokes a Service's onCreate() & onStartCommand() hook methods

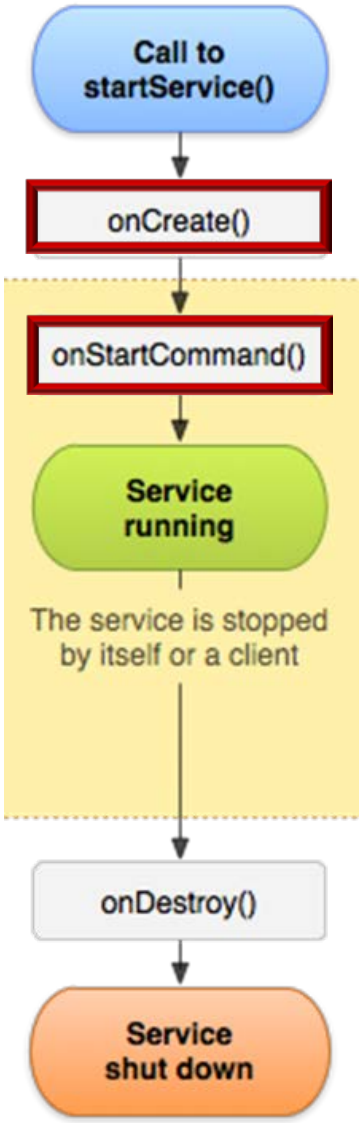



```
public class DownloadService extends Service {
  public void onCreate() { ... }
  public int onStartCommand(Intent intent,
             int flags, int startId) { ... }
  ...
```

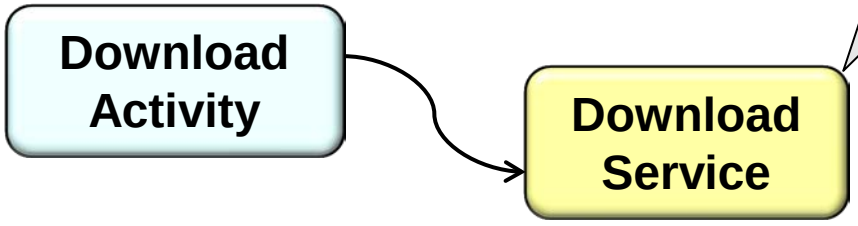


Processing a Started Service

- Android launches the Service if it's not already running
- Started Services are driven by "inversion of control"
  - e.g., the Android Service framework invokes a Service's onCreate() & onStartCommand() hook methods




```
public class DownloadService extends Service {
  public void onCreate() { ... }
  public int onStartCommand(Intent intent,
             int flags, int startId) { ... }
  ...
```



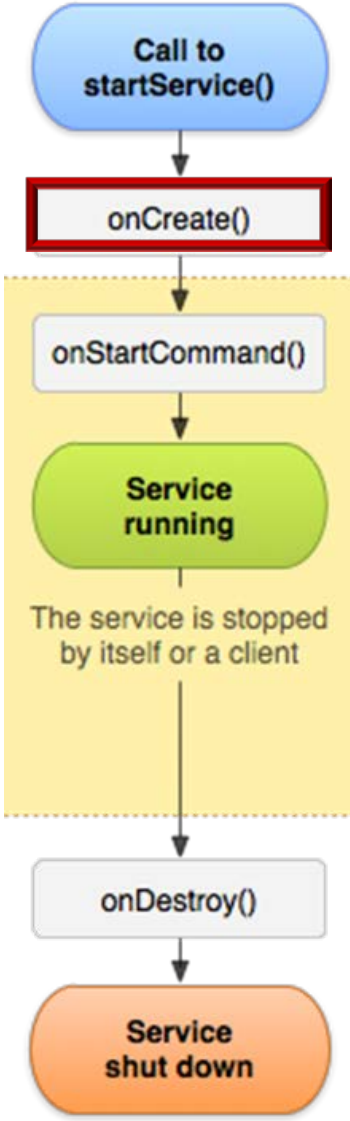
Processing a Started Service

- Android launches the Service if it's not already running
- Started Services are driven by "inversion of control"
  - e.g., the Android Service framework invokes a Service's onCreate() & onStartCommand() hook methods

onCreate() starts a HandlerThread

```
public class DownloadService extends Service {
  public void onCreate() {
    ...
  }
```

Download Activity → **Download Service**



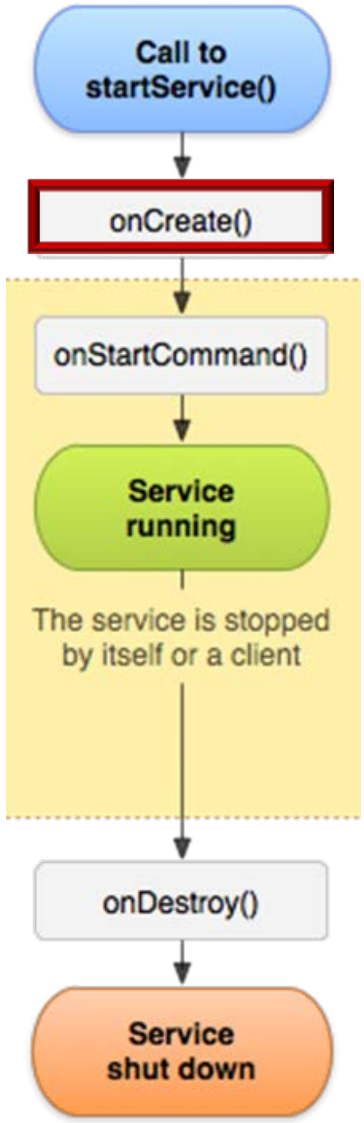
Processing a Started Service

- Android launches the Service if it's not already running
- Started Services are driven by "inversion of control"
  - e.g., the Android Service framework invokes a Service's onCreate() & onStartCommand() hook methods

HandlerThread works with a ServiceHandler to download the image in the background & return pathname to client

```
public class DownloadService extends Service {
  public void onCreate() {
    ...
  }
```

Download Activity → **Download Service**



Processing a Started Service (Part 2)

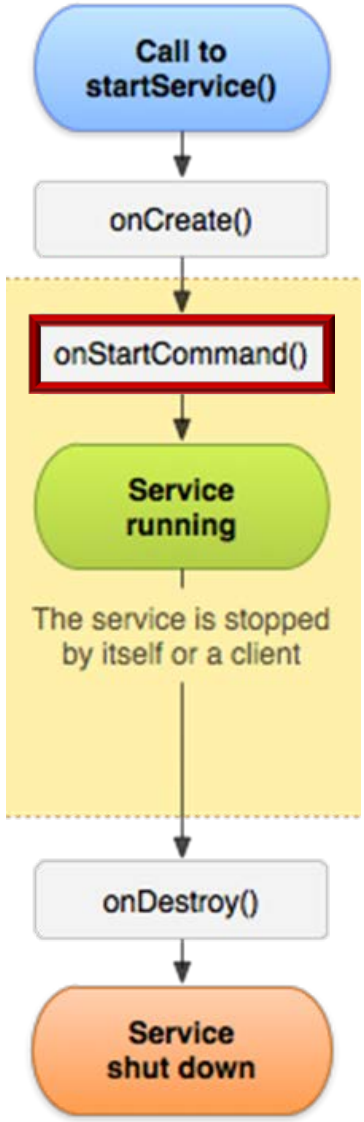
Processing a Started Service

- Android launches the Service if it's not already running
- Started Services are driven by "inversion of control"
  - e.g., the Android Service framework invokes a Service's onCreate() & onStartCommand() hook methods

onStartCommand() sends the Intent to the background HandlerThread

```
public class DownloadService extends Service {
  public int onStartCommand(Intent intent,
                            int flags,
                            int startId)
  { return ...; }
```

Download Activity → **Download Service**



Processing a Started Service

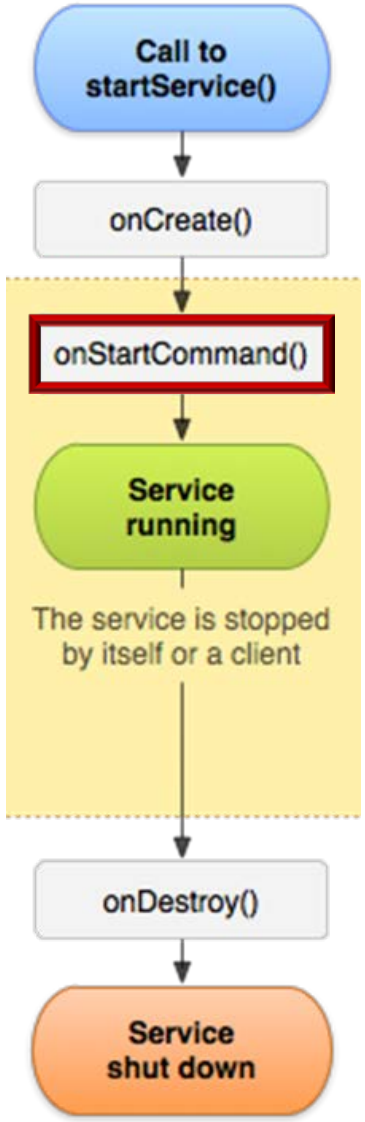
- Android launches the Service if it's not already running
- Started Services are driven by "inversion of control"
  - e.g., the Android Service framework invokes a Service's onCreate() & onStartCommand() hook methods

onStartCommand() returns a result to Android, but not to the client

```
public class DownloadService extends Service {
  public int onStartCommand(Intent intent,
                            int flags,
                            int startId)
  { return ...; }
```

Download Activity

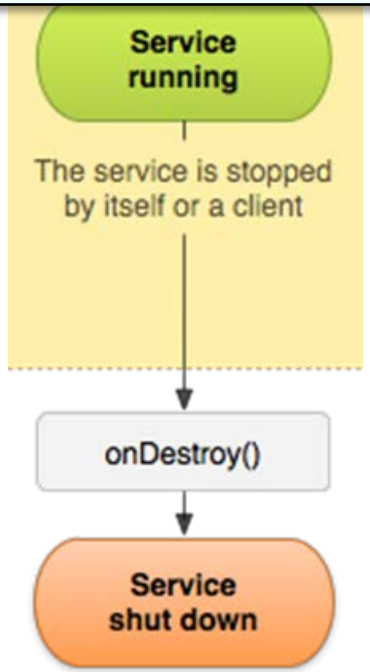
Download Service

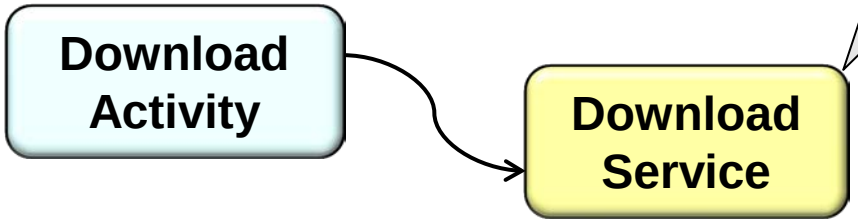


Processing a Started Service

Return value tells Android what it should do with the Service if its process is killed while it is running

```
public class DownloadService extends Service {
  public int onStartCommand(Intent intent,
                            int flags,
                            int startId)
  { return ...; }
```



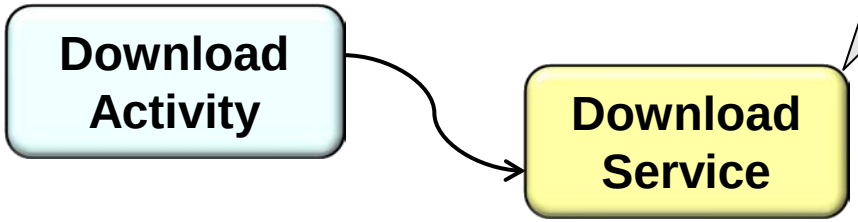


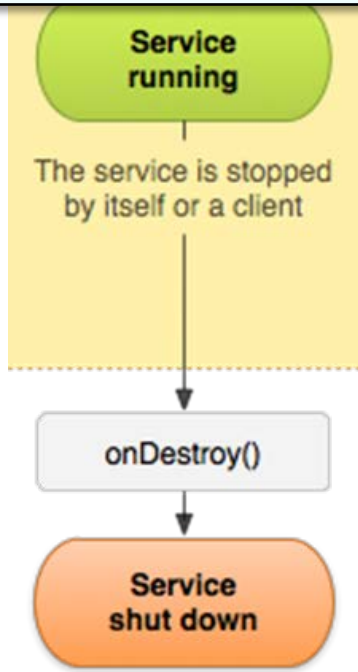
Processing a Started Service

Return value tells Android what it should do with the Service if its process is killed while it is running

- *START_STICKY – Don't redeliver Intent to onStartCommand() (pass null intent)*

```
public class DownloadService extends Service {
  public int onStartCommand(Intent intent,
                            int flags,
                            int startId)
  { return ...; }
```



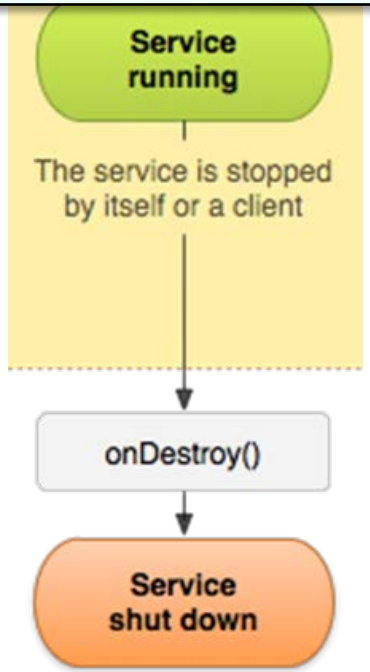


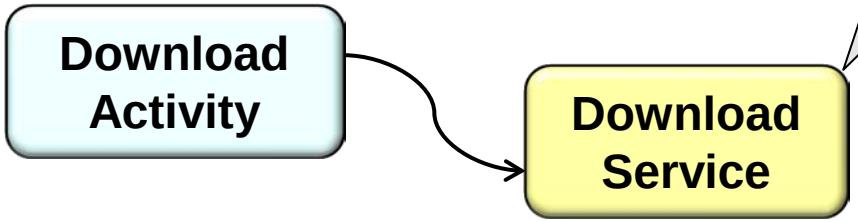
Processing a Started Service

Return value tells Android what it should do with the Service if its process is killed while it is running

- *START_STICKY – Don't redeliver Intent to onStartCommand() (pass null intent)*
- *START_NOT_STICKY – Service should remain stopped until explicitly started by some client code*

```
public class DownloadService extends Service {
  public int onStartCommand(Intent intent,
                            int flags,
                            int startId)
  { return ...; }
```



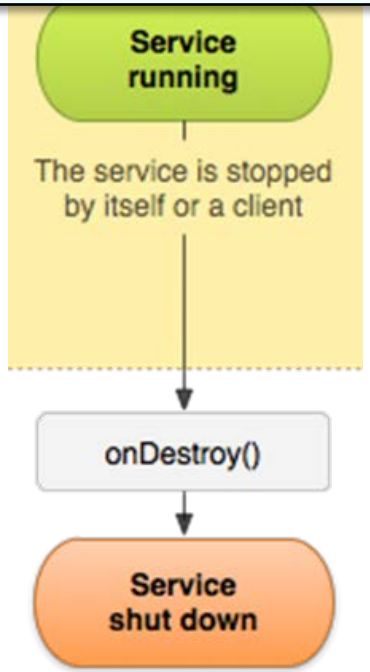


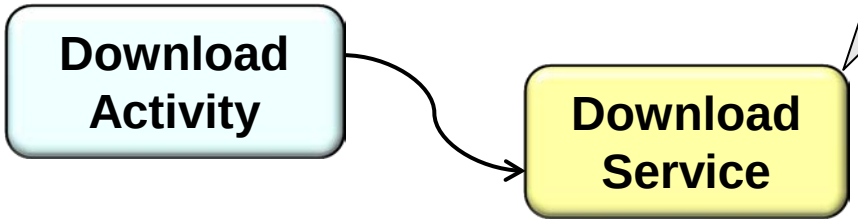
Processing a Started Service

Return value tells Android what it should do with the Service if its process is killed while it is running

- *START_STICKY – Don't redeliver Intent to onStartCommand() (pass null intent)*
- *START_NOT_STICKY – Service should remain stopped until explicitly started by some client code*
- *START_REDELIVER_INTENT – Restart Service via onStartCommand(), supplying the same Intent as was delivered this time*

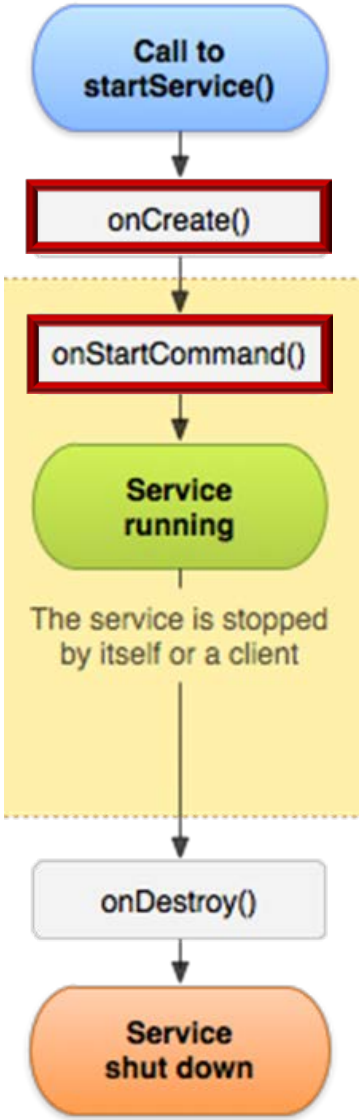
```
public class DownloadService extends Service {
  public int onStartCommand(Intent intent,
                            int flags,
                            int startId)
  { return ...; }
```



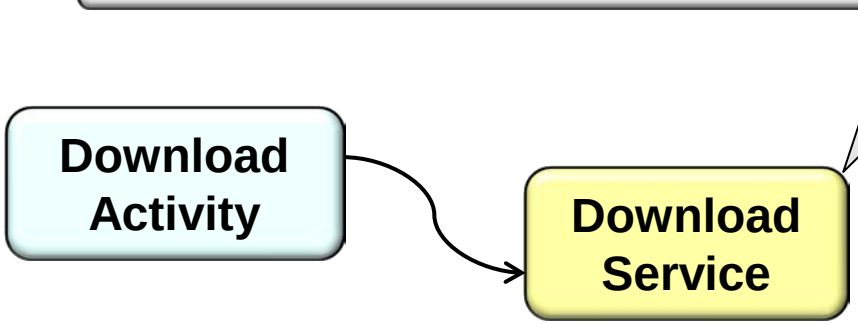


Processing a Started Service

- Android launches the Service if it's not already running
- Started Services are driven by "inversion of control"
  - e.g., the Android Service framework invokes a Service's onCreate() & onStartCommand() hook methods



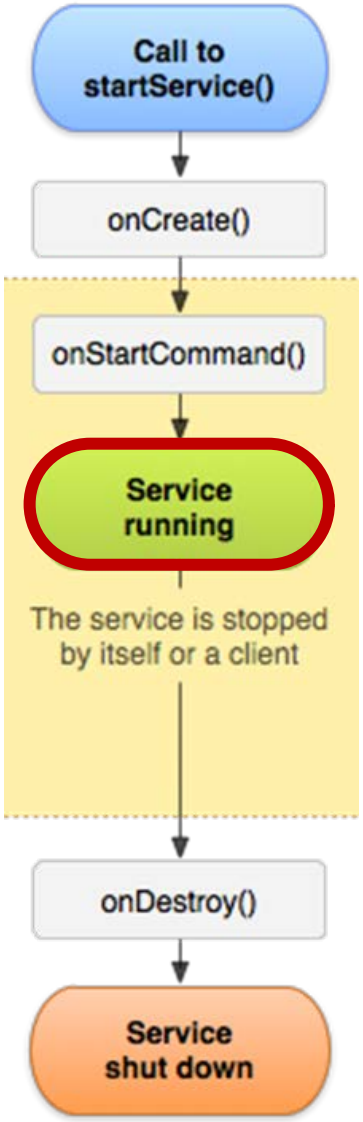

```
public class DownloadService extends Service {
  public int onStartCommand(Intent intent,
                            int flags,
                            int startId)
  { return ...; }
```



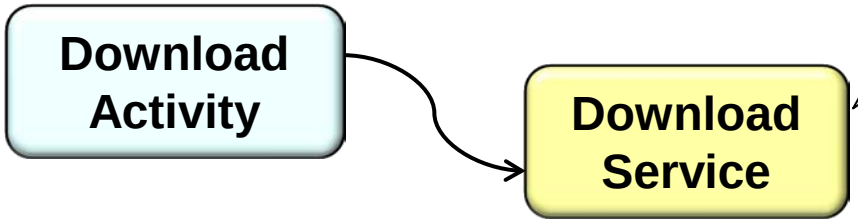
See android-developers.blogspot.com.au/2010/02/service-api-changes-starting-with.html

Processing a Started Service

- Android launches the Service if it's not already running
- Started Services are driven by "inversion of control"
- A started service typically performs a single operation & often doesn't return a result to the client




```
public class DownloadService extends Service {
  ...
  public void handleMessage(Message msg) {
      downloadImageAndReply((Intent) msg.obj);
      ...
```



Processing a Started Service

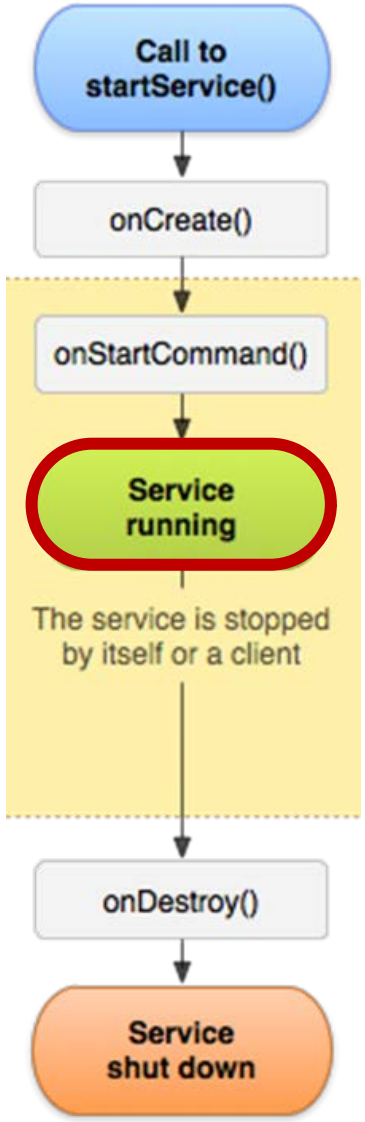
- Android launches the Service if it's not already running
- Started Services are driven by "inversion of control"
- A started service typically performs a single operation & often doesn't return a result to the client

Retrieve an image from the remote server & return pathname to client

```
public class DownloadService extends Service {
  ...
  public void handleMessage(Message msg) {
      downloadImageAndReply((Intent) msg.obj);
      ...
```

Download Activity

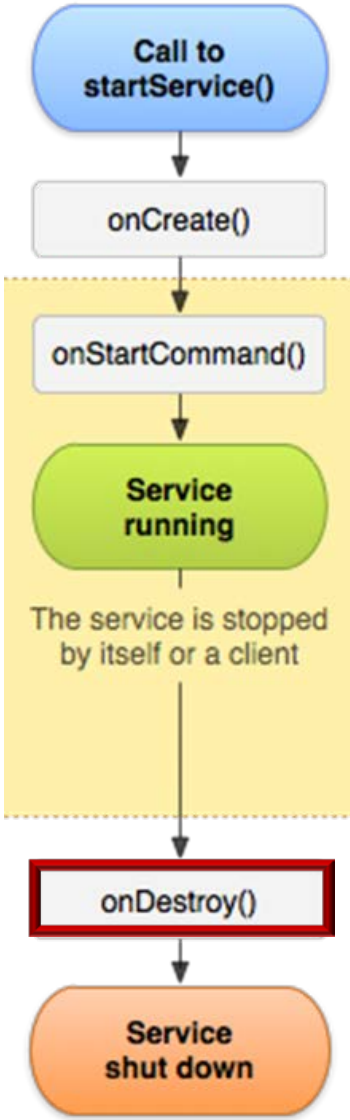
Download Service

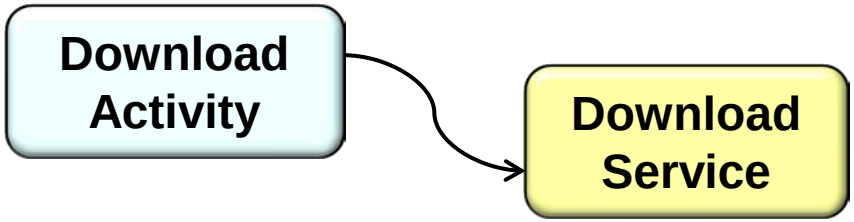


Stopping a Started Service

Stopping a Started Service

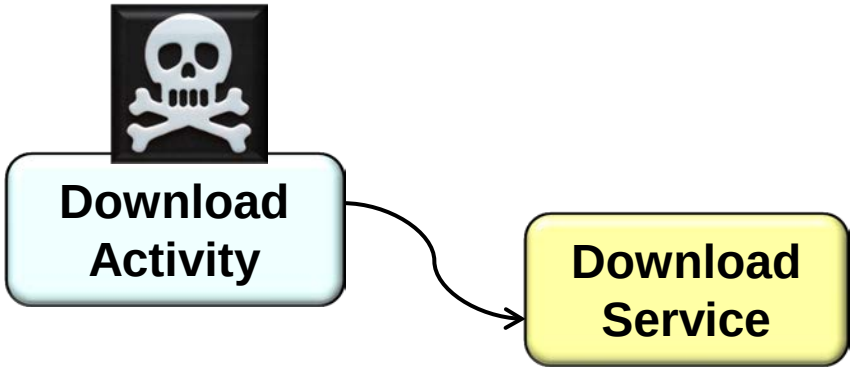
- A Started Service's lifecycle is independent of the component that launched it

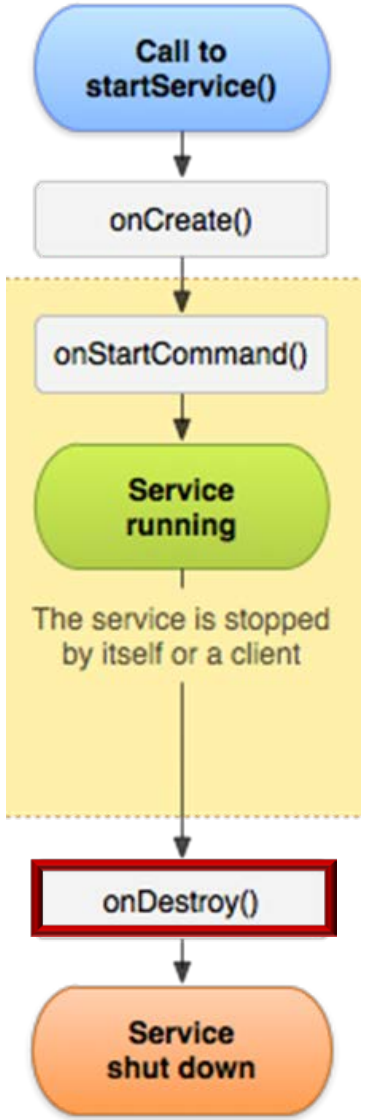




Stopping a Started Service

- A Started Service's lifecycle is independent of the component that launched it
  - i.e., it can run in the background indefinitely, even if the component that launched it is destroyed

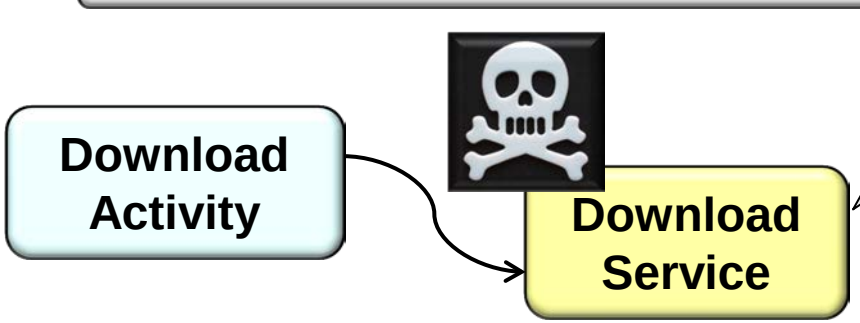


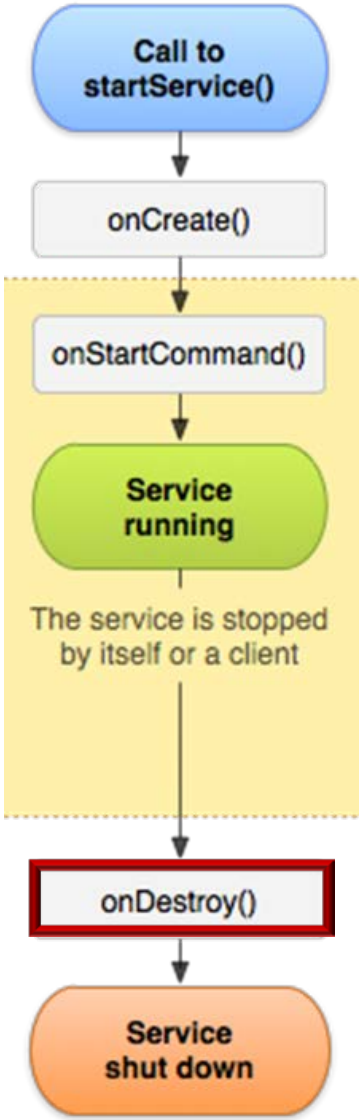


Stopping a Started Service

- A Started Service's lifecycle is independent of the component that launched it
- A Service must be stopped when its operation is done

```
public class DownloadService extends Service {
  ...
  public void handleMessage(Message msg) {
      ...
      stopSelf(msg.arg1);
```





Stopping a Started Service

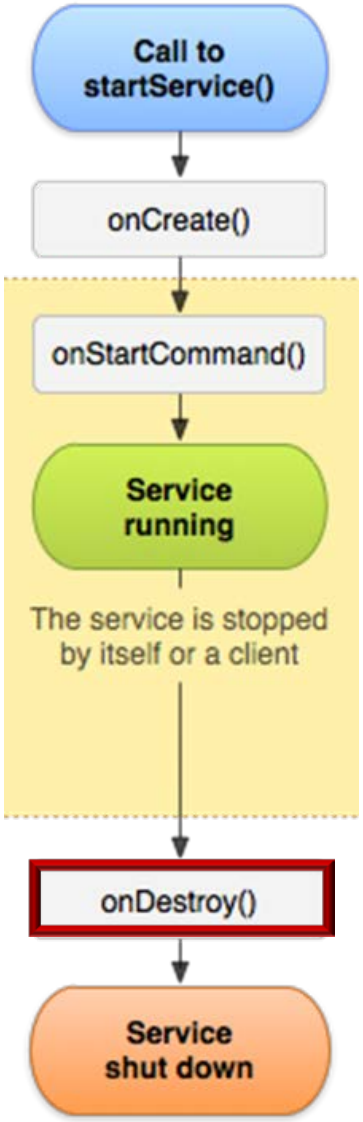
- A Started Service's lifecycle is independent of the component that launched it
- A Service must be stopped when its operation is done, e.g.
  - It can call stopSelf() to shut itself down

A Service that stops itself must be careful there aren't concurrent operations processing other Intents!

```
public class DownloadService extends Service {
  ...
  public void handleMessage(Message msg) {
      ...
      stopSelf(msg.arg1);
```

Download Activity

Download Service



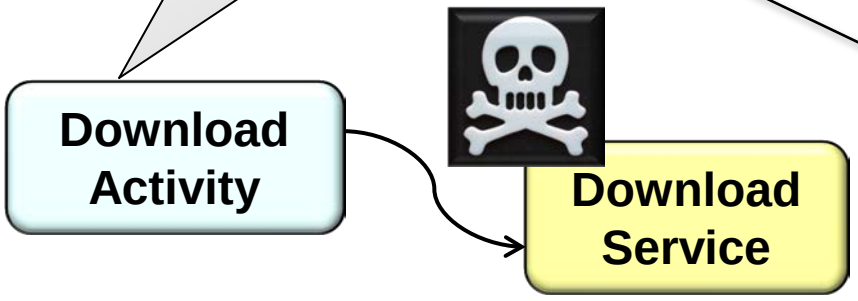
See upcoming discussion about the Android "Concurrent Service Stopping" idiom

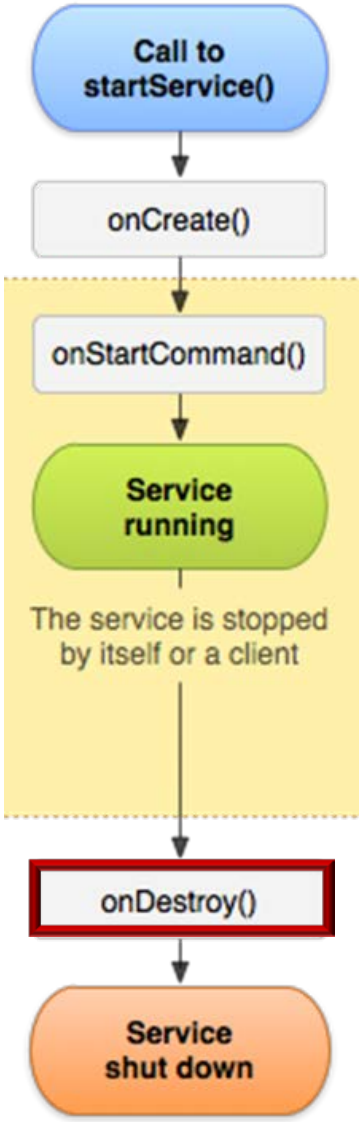
Stopping a Started Service

- A Started Service's lifecycle is independent of the component that launched it
- A Service must be stopped when its operation is done, e.g.
  - It can call stopSelf() to shut itself down
  - Another component can shut down a Service by calling stopService()

```
public class DownloadActivity ... {
  ...
  stopService(intent);
  ...
```

If Service isn't running, nothing happens





Calls to startService() are not counted, so the Service is stopped no matter how many times it was started