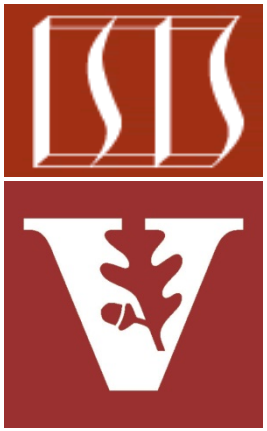
Android Services & Security: Programming Started Services (Part 1)

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



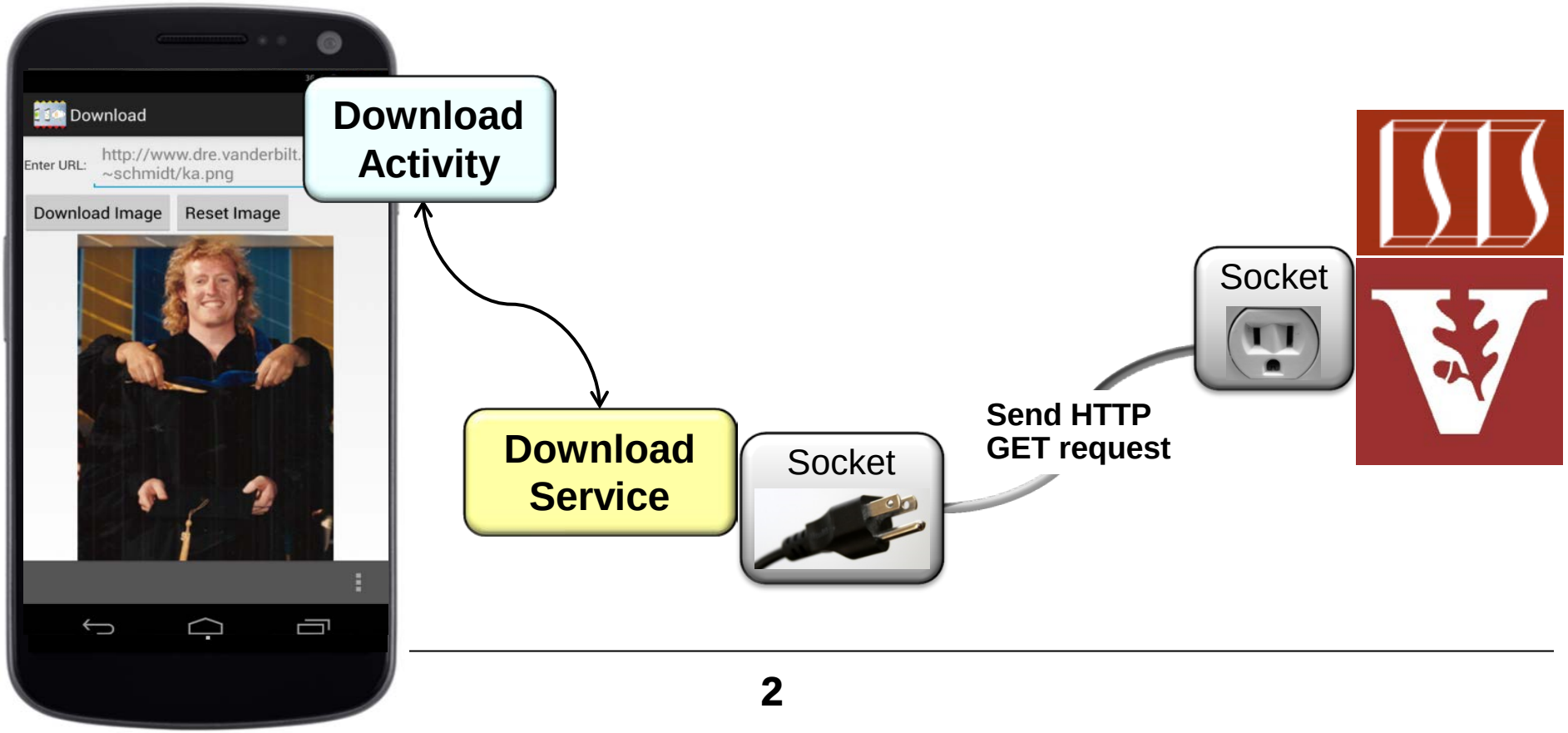
Professor of Computer Science

Institute for Software Integrated Systems

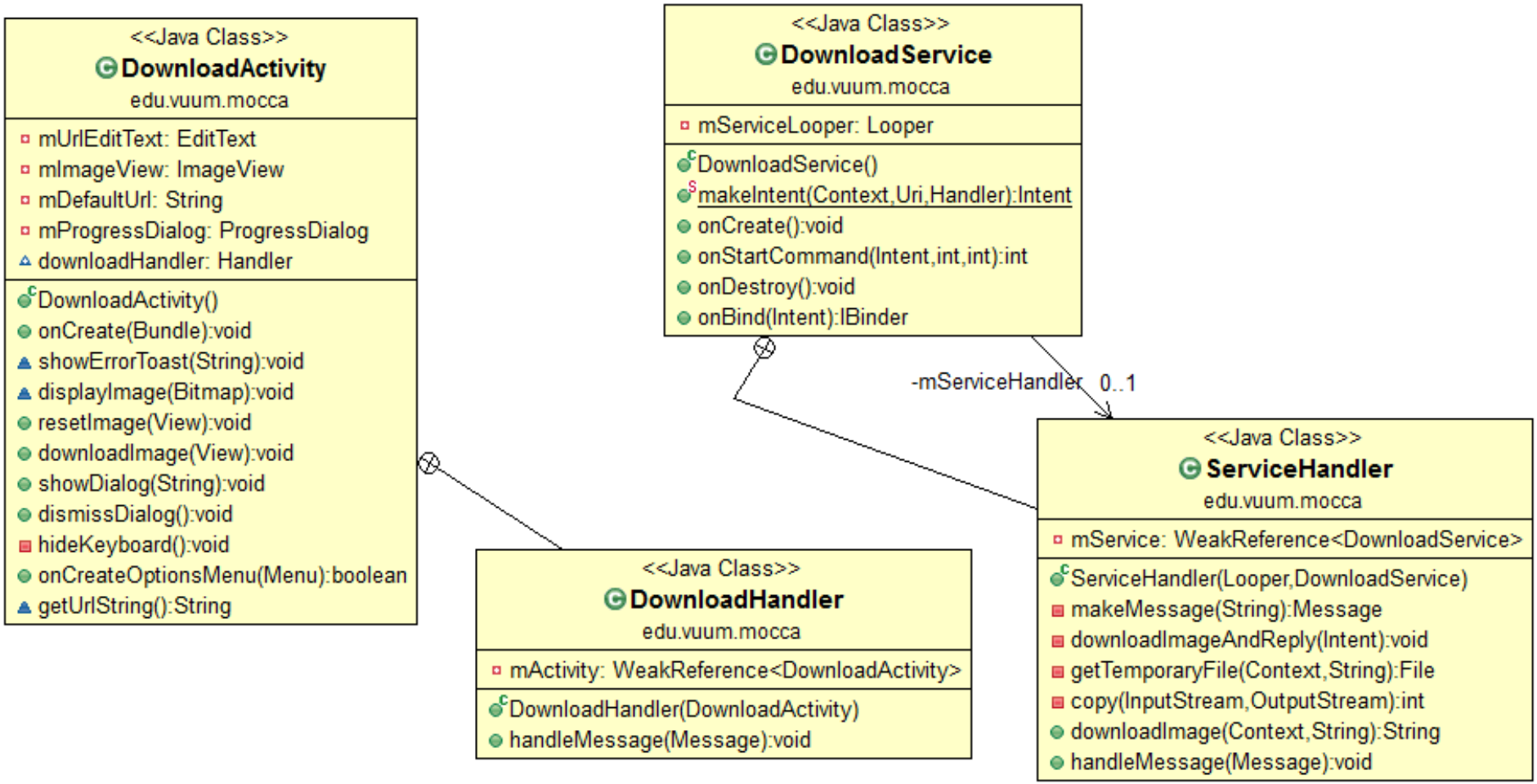
Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Module

- Understand how to program a Started Service
  - e.g., a Download Application that uses a Started Service to retrieve & display an image from a remote server

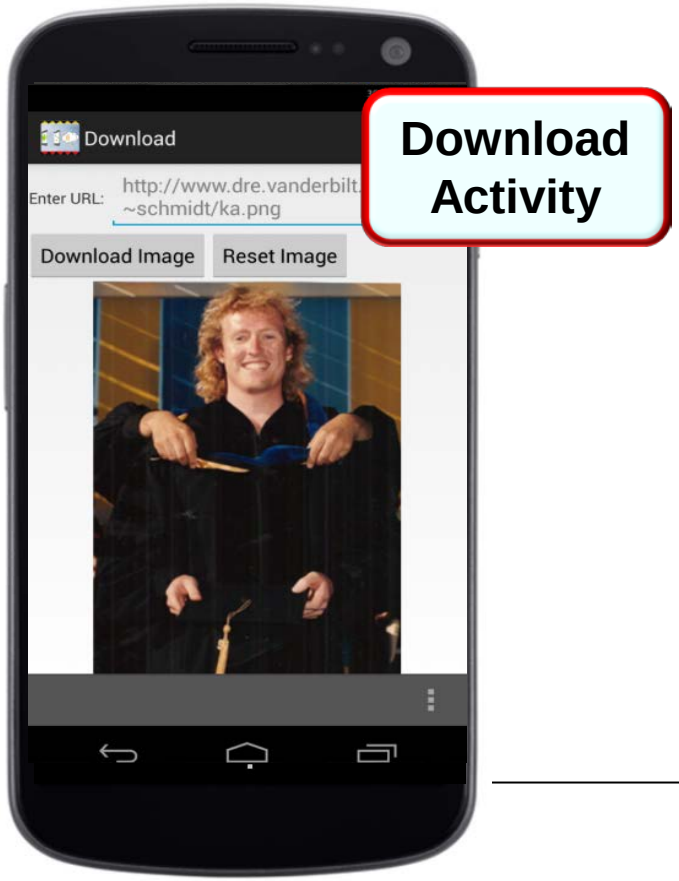


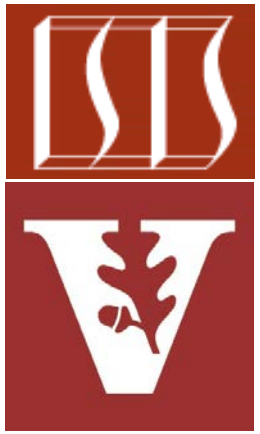
Download Application Overview



Download Application Overview

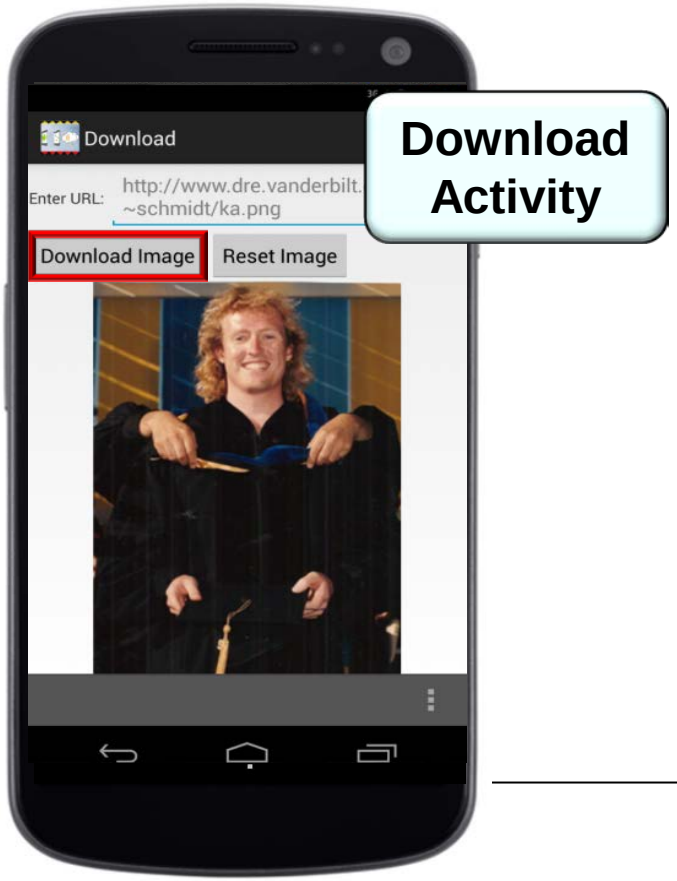
1. DownloadActivity uses startService() to launch a DownloadService

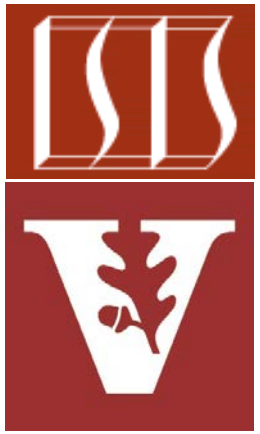




Download Application Overview

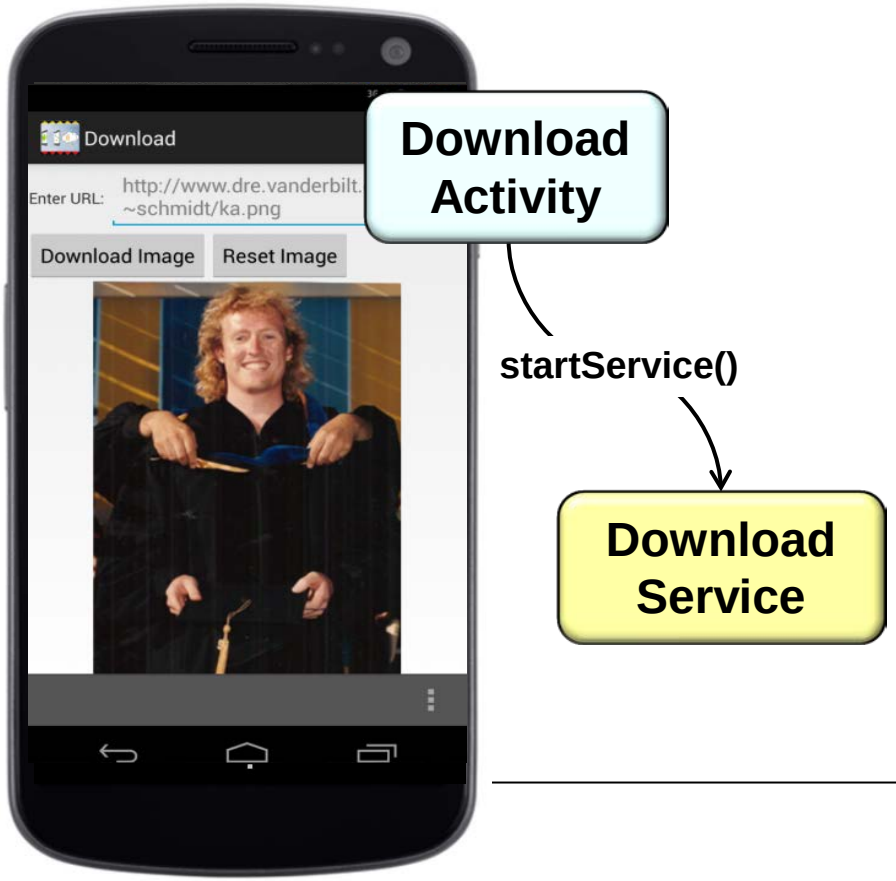
1. DownloadActivity uses startService() to launch a DownloadService

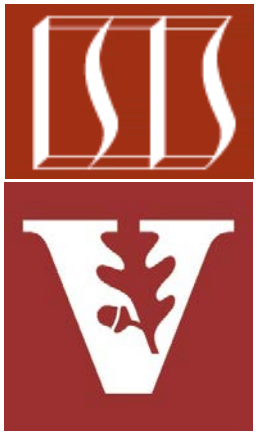




Download Application Overview

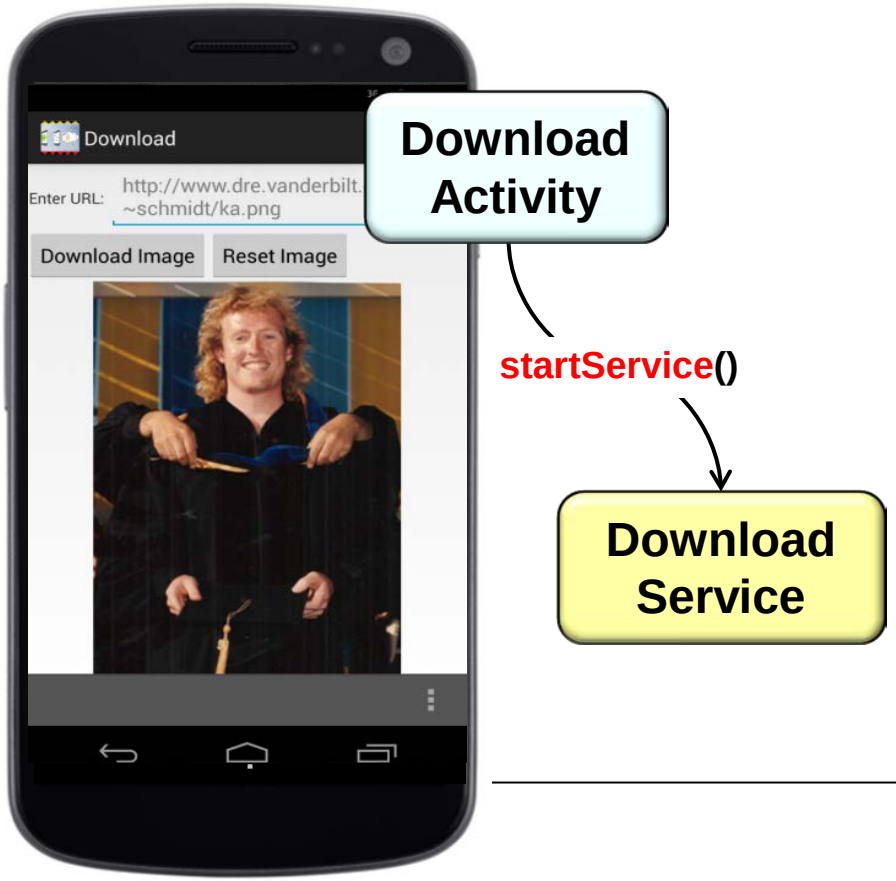
1. DownloadActivity uses startService() to launch a DownloadService

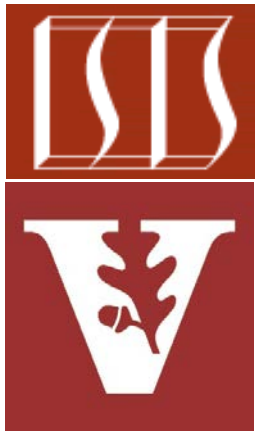




Download Application Overview

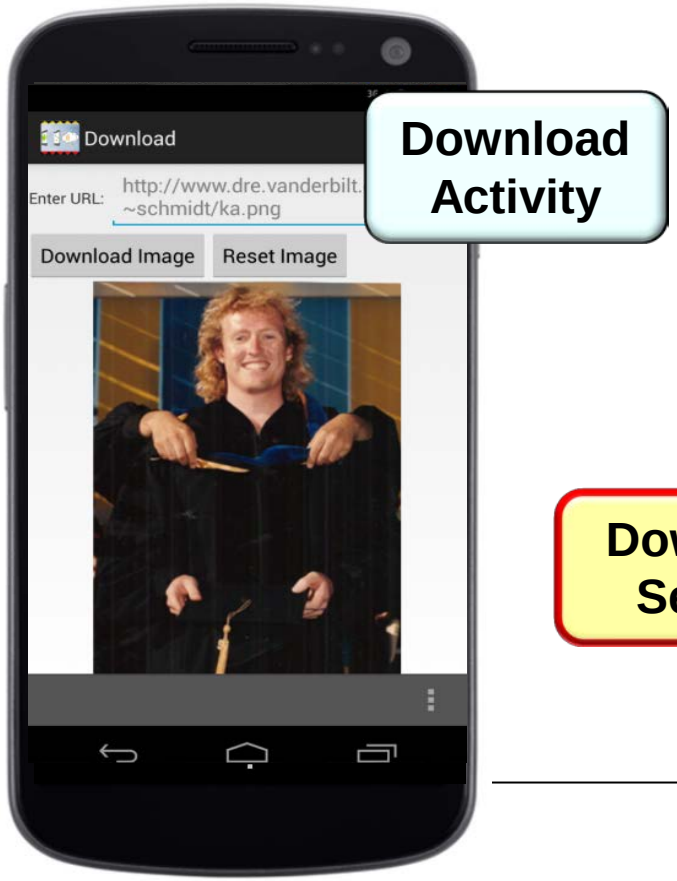
1. DownloadActivity uses startService() to launch a DownloadService

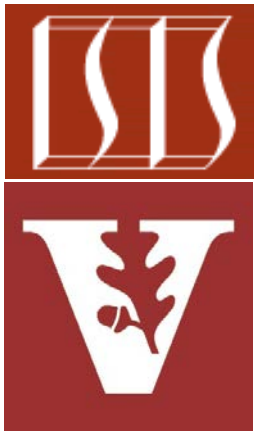




Download Application Overview

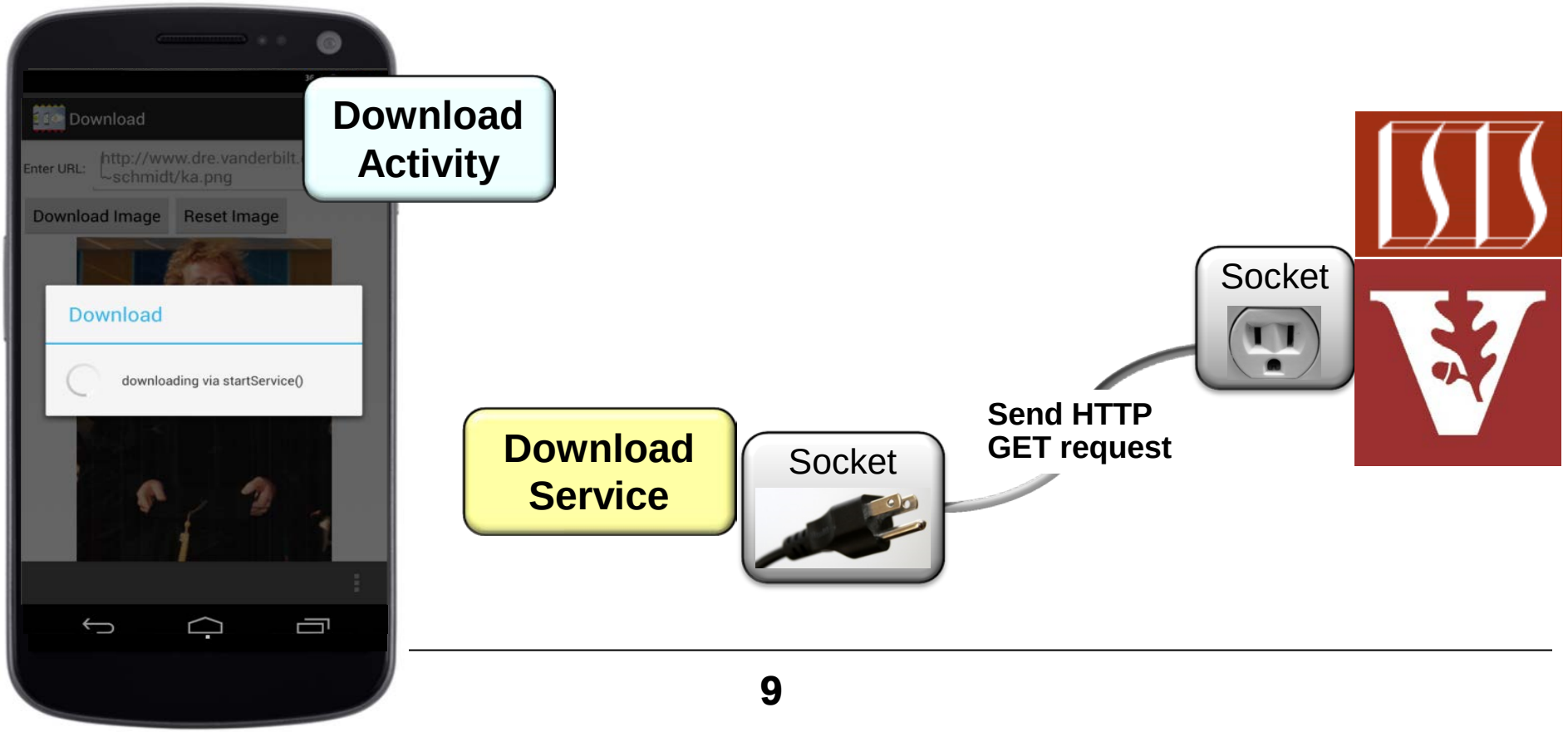
1. DownloadActivity uses startService() to launch a DownloadService
2. The DownloadService retrieves the image & stores it in a file on the device





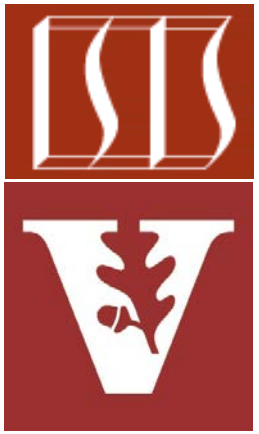
Download Application Overview

1. DownloadActivity uses startService() to launch a DownloadService
2. The DownloadService retrieves the image & stores it in a file on the device



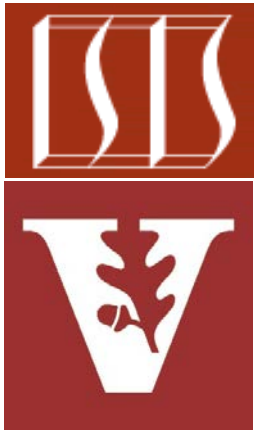
Download Application Overview

1. DownloadActivity uses startService() to launch a DownloadService
2. The DownloadService retrieves the image & stores it in a file on the device
3. The DownloadService returns the pathname of the file back to the DownloadActivity, which then displays the image



Download Application Overview

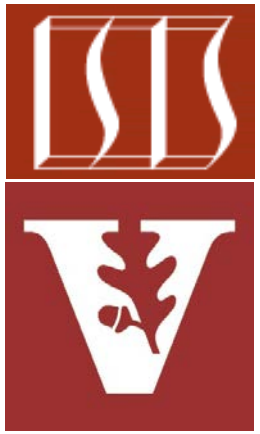
1. DownloadActivity uses startService() to launch a DownloadService
2. The DownloadService retrieves the image & stores it in a file on the device
3. The DownloadService returns the pathname of the file back to the DownloadActivity, which then displays the image



We'll just cover steps 1 & 2 in this part

Download Application Overview

1. DownloadActivity uses startService() to launch a DownloadService
2. The DownloadService retrieves the image & stores it in a file on the device
3. The DownloadService returns the pathname of the file back to the DownloadActivity, which then displays the image



See upcoming parts on “Activity & Service Communication”

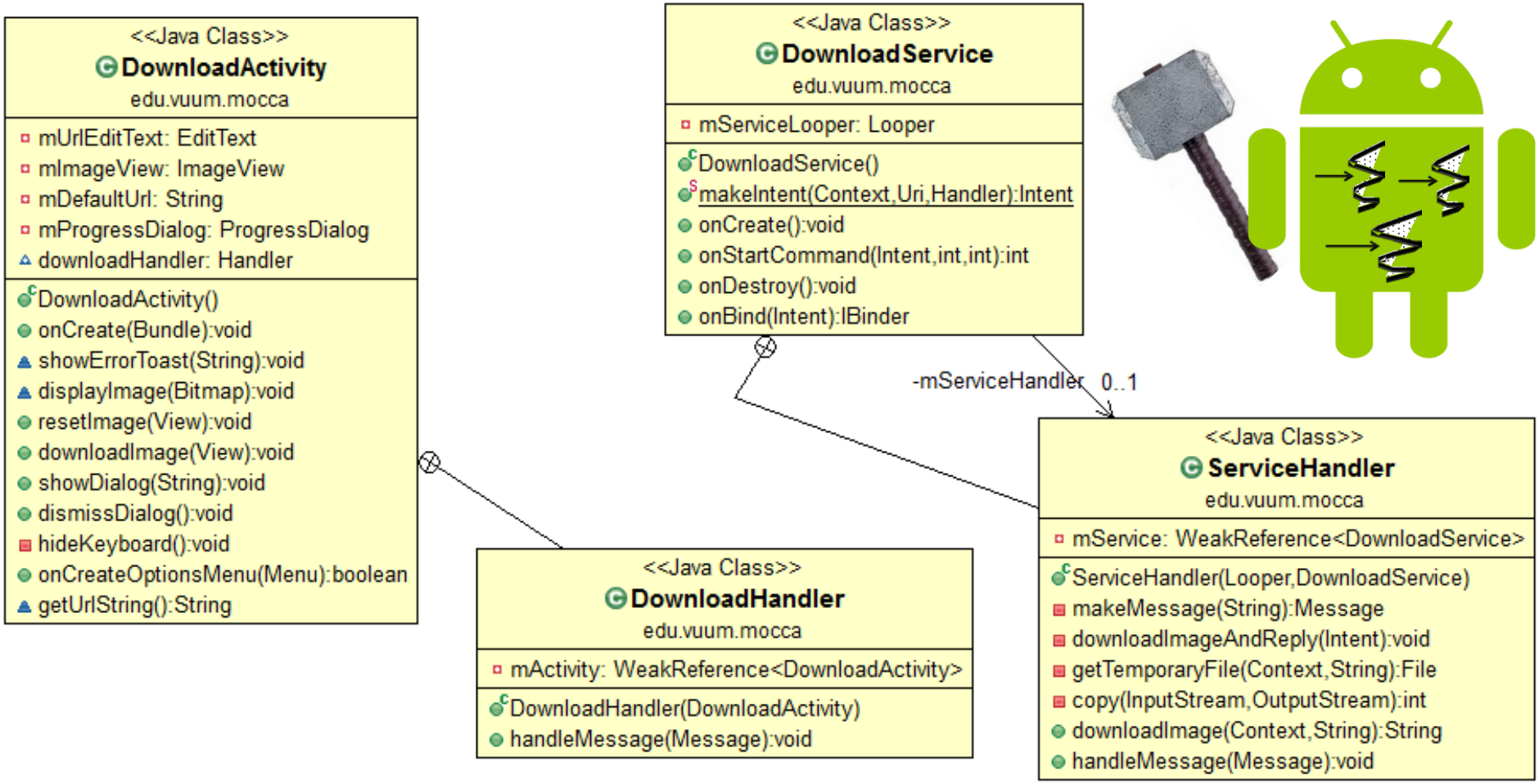
Tips to Understand the Download Application

- Run/read the code & watch the video carefully to understand how it works

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

Tips to Understand the Download Application

- Run/read the code & watch the video carefully to understand how it works
- This example is complex since many classes & Android mechanisms are used

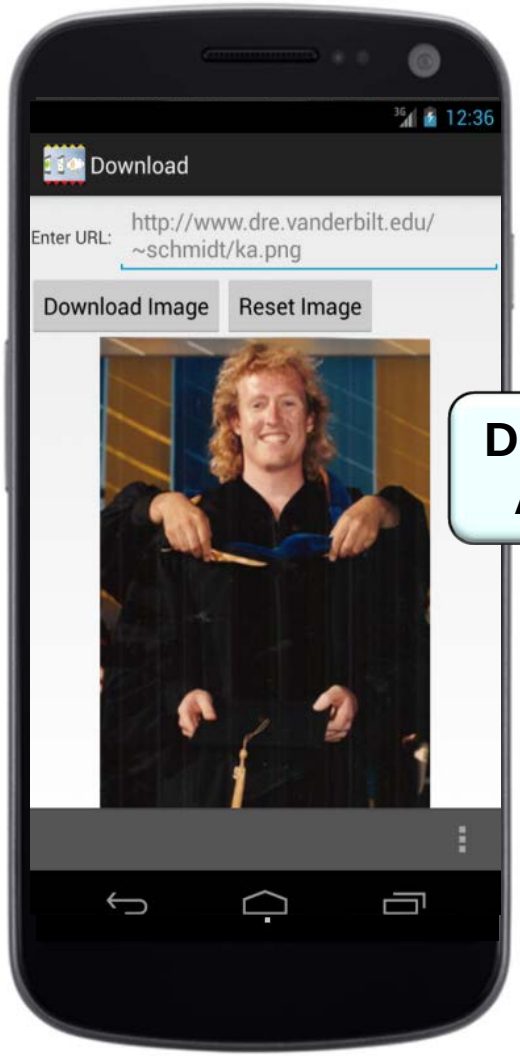


Tips to Understand the Download Application

- Run/read the code & watch the video carefully to understand how it works
- This example is complex since many classes & Android mechanisms are used
  - We therefore analyze it from various perspectives

Programming Started Services (Part 1)

Programming a Started Service



Download Activity

Download Service

Socket

Socket

Send HTTP GET request

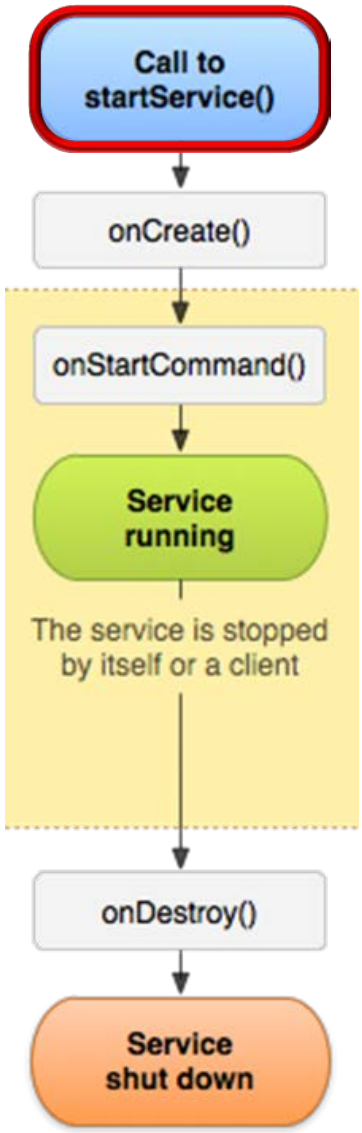
Launching a Started Service

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

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

startService(intent);
```

Download Activity



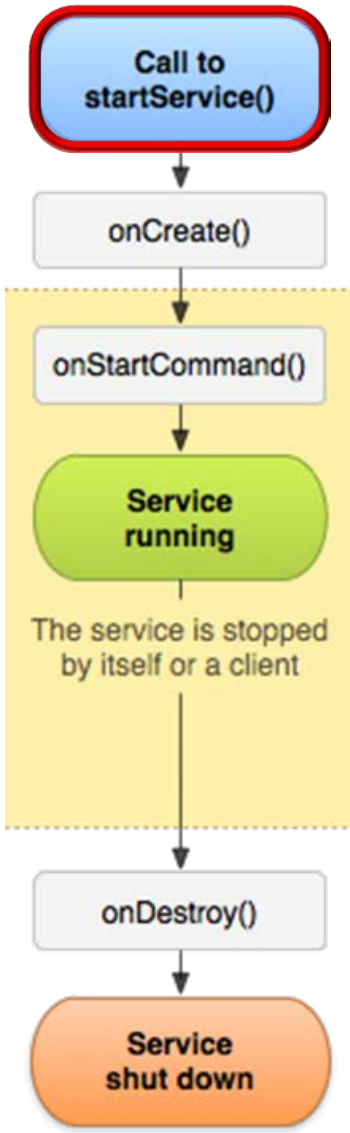
Launching a Started Service

- A client launches a Started Service by calling startService()
  - e.g., Download Activity 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



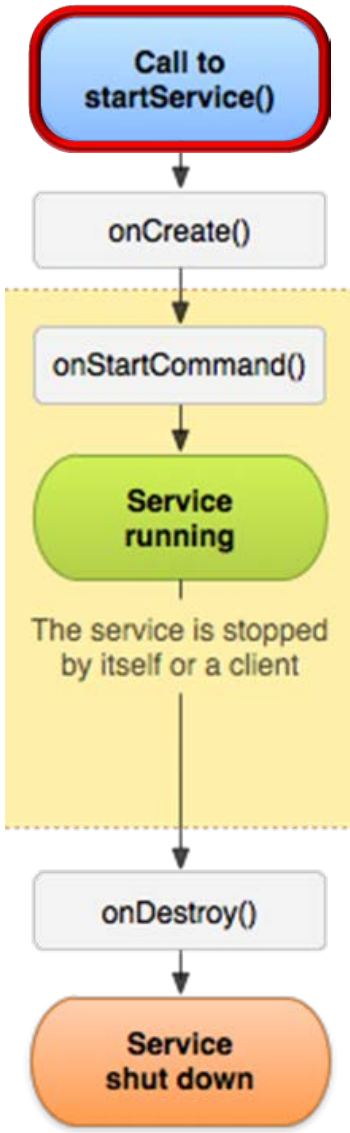
Launching a Started Service

- A client launches a Started Service by calling startService()
  - e.g., Download Activity 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

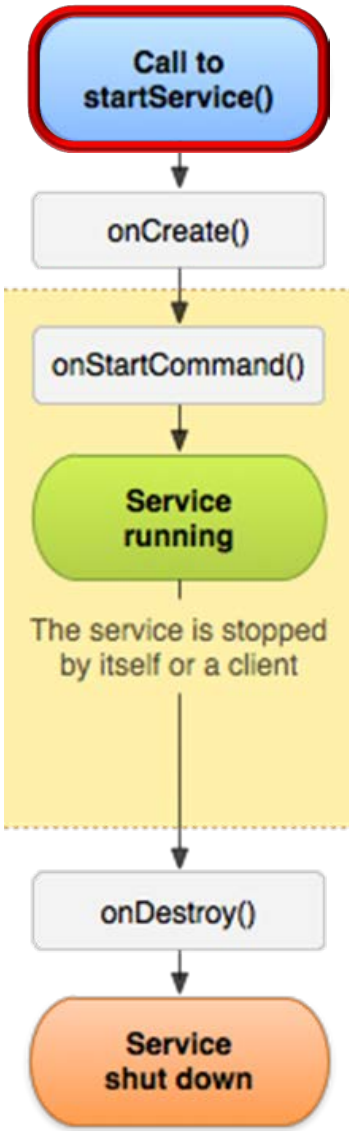
- A client launches a Started Service by calling startService()
  - e.g., Download Activity creates an Intent that identifies the DownloadService & supplies a URL parameter via Intent data that tells Service what image to retrieve

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

startService(intent);
```

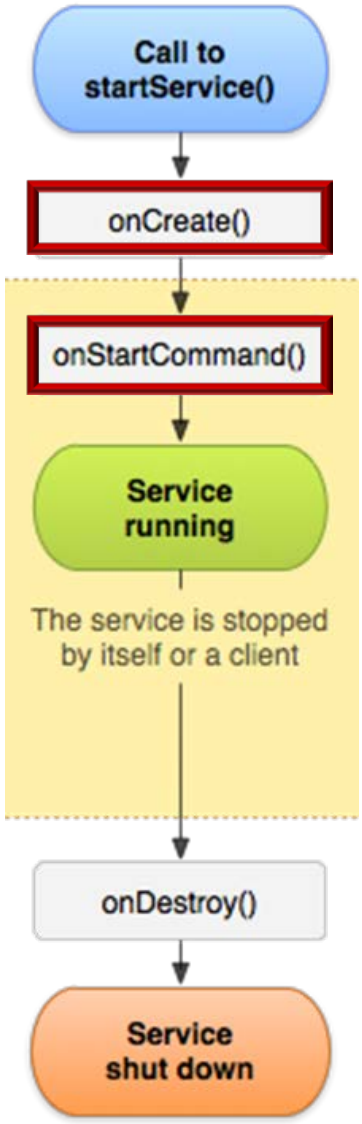
Download Activity

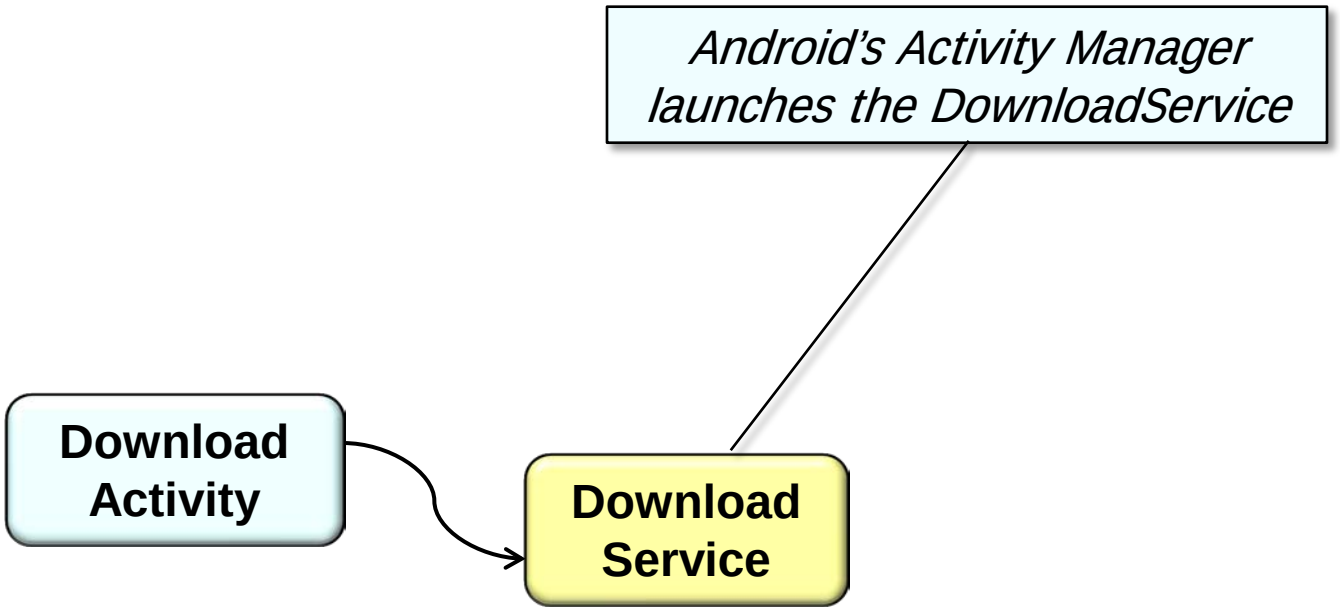
This call doesn't block the client while the DownloadService runs



Processing a Started Service

- In response to startService(), Android will launch the Service if it's not already running

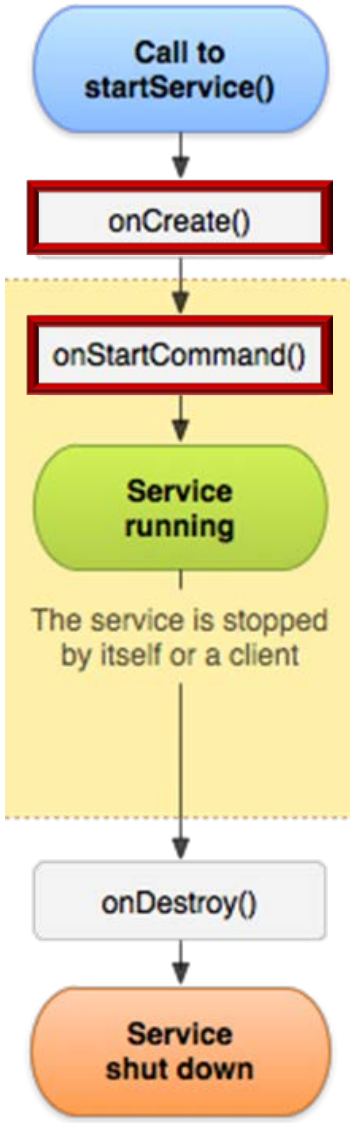




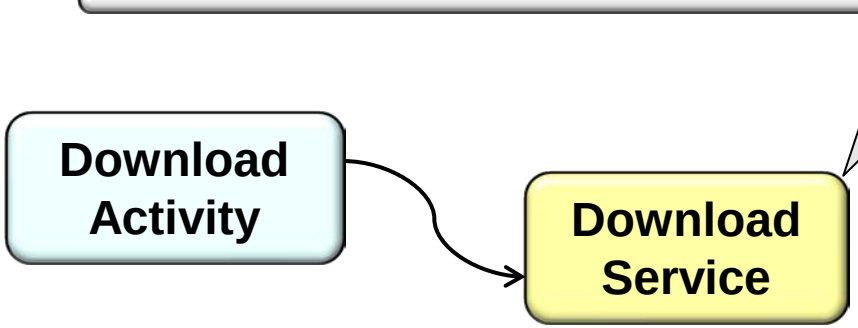
www.dre.vanderbilt.edu/~schmidt/PDF/Activator.pdf has more info

Processing a Started Service

- In response to startService(), Android will launch the Service if it's not already running
- Android then invokes the Service's onCreate() & onStartCommand() hook methods

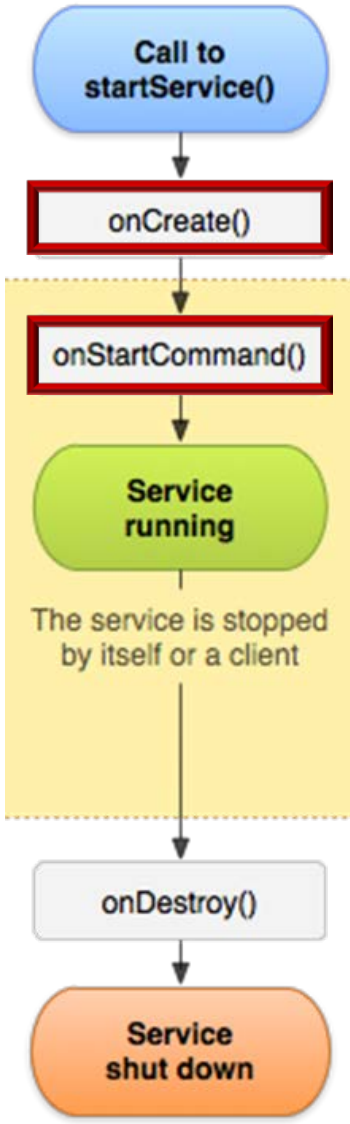



```
public class DownloadService extends Service {
  public void onCreate() { ... }
  public int onStartCommand(Intent intent,
             int flags, int startId) { ... }
  ...
```

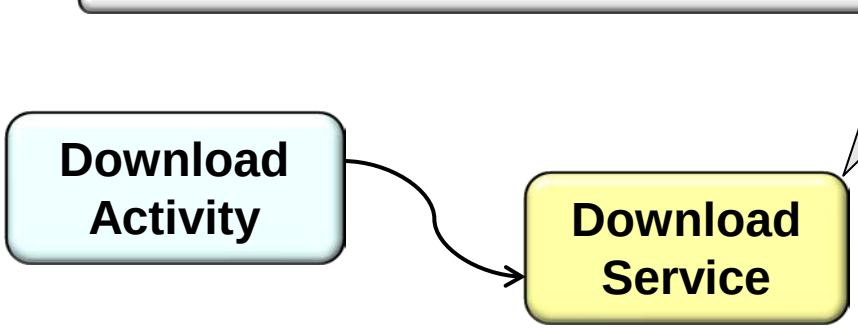


Processing a Started Service

- In response to startService(), Android will launch the Service if it's not already running
- Android then invokes the Service's onCreate() & onStartCommand() hook methods




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



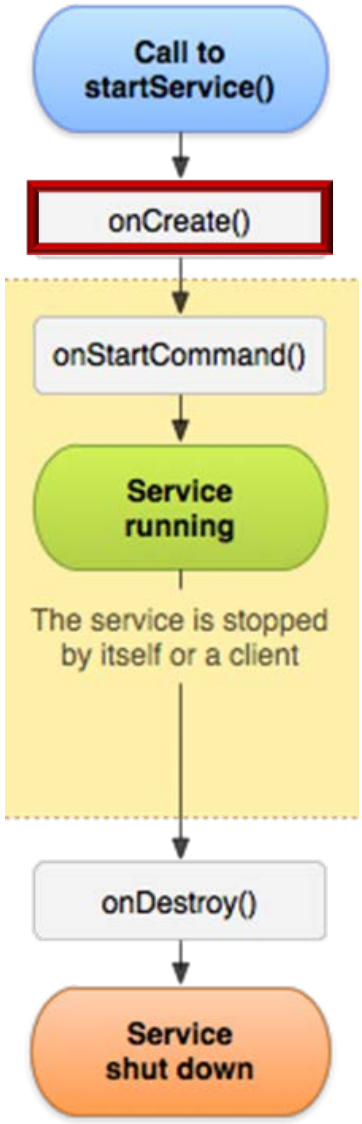
Processing a Started Service

- In response to startService(), Android will launch the Service if it's not already running
- Android then invokes the 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

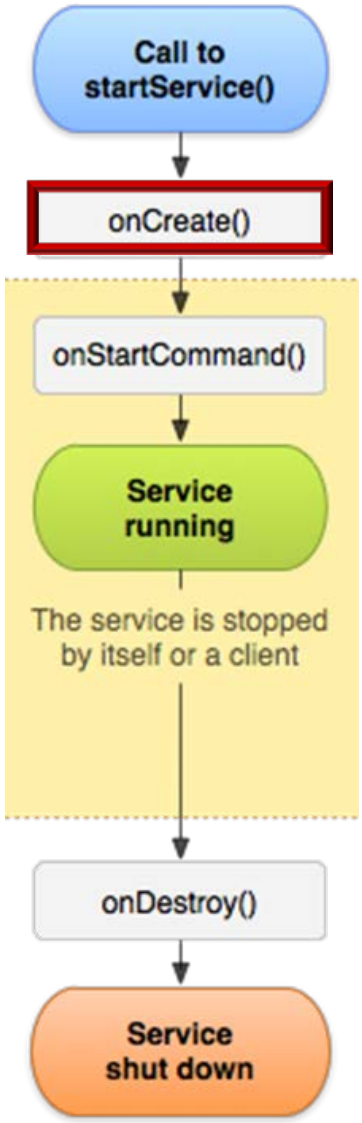
- In response to startService(), Android will launch the Service if it's not already running
- Android then invokes the Service's onCreate() & onStartCommand() hook methods

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



Programming Started Services (Part 2)

Processing a Started Service

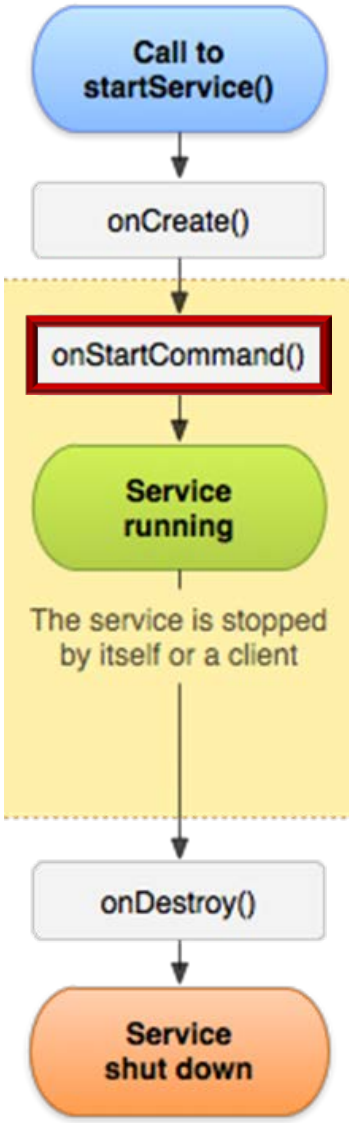
- In response to startService(), Android will launch the Service if it's not already running
- Android then invokes the Service's onCreate() & onStartCommand() hook methods

onStartCommand() sends the Intent to the background HandlerThread

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

  { ... }
```

Download Activity → **Download Service**



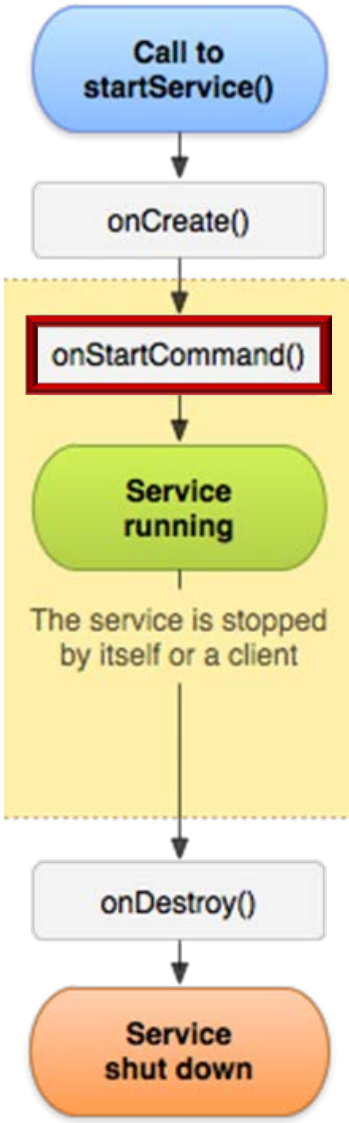
Processing a Started Service

- In response to startService(), Android will launch the Service if it's not already running
- Android then invokes the 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 ...; }
```

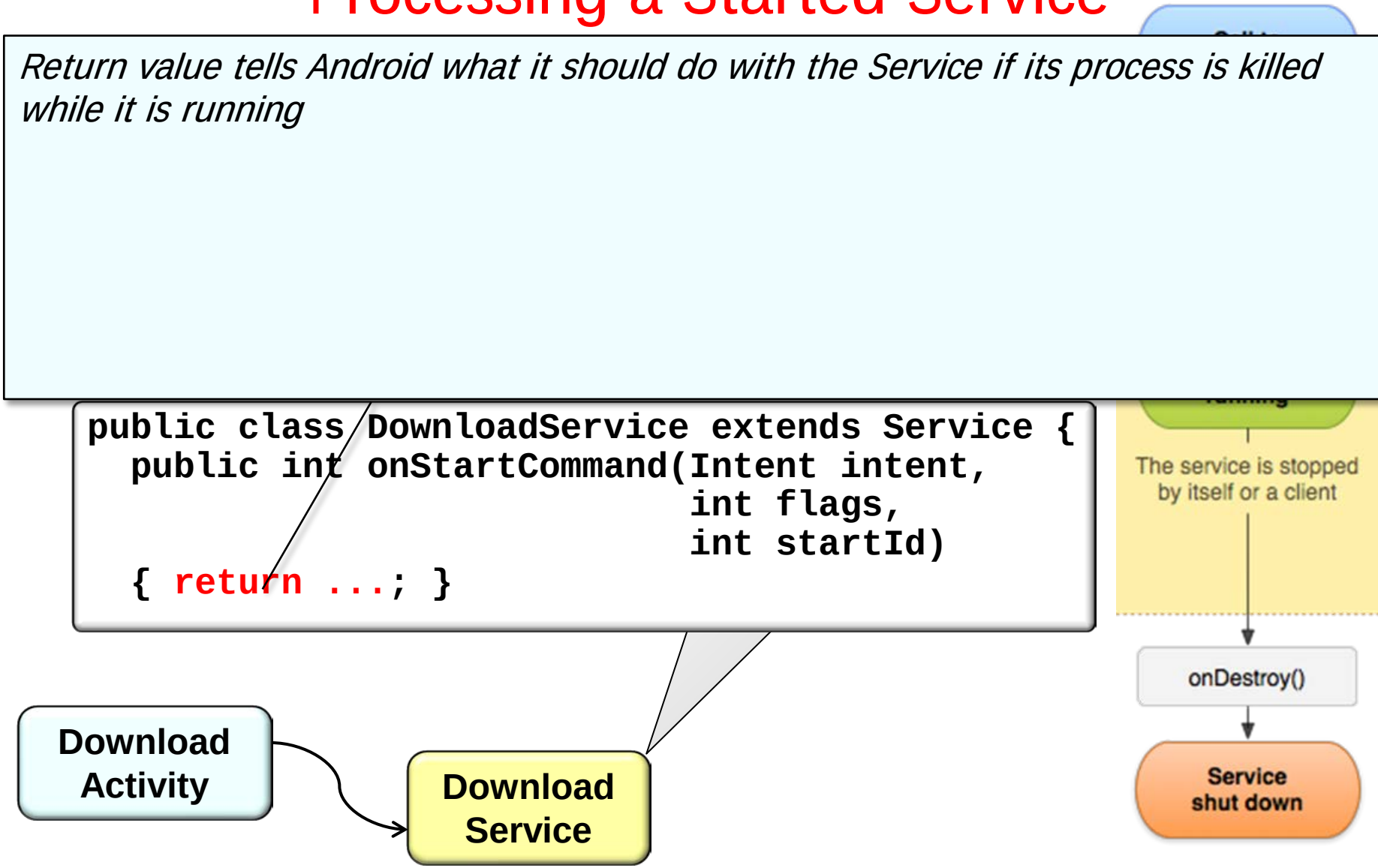
Download Activity → **Download Service**



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 ...; }
```

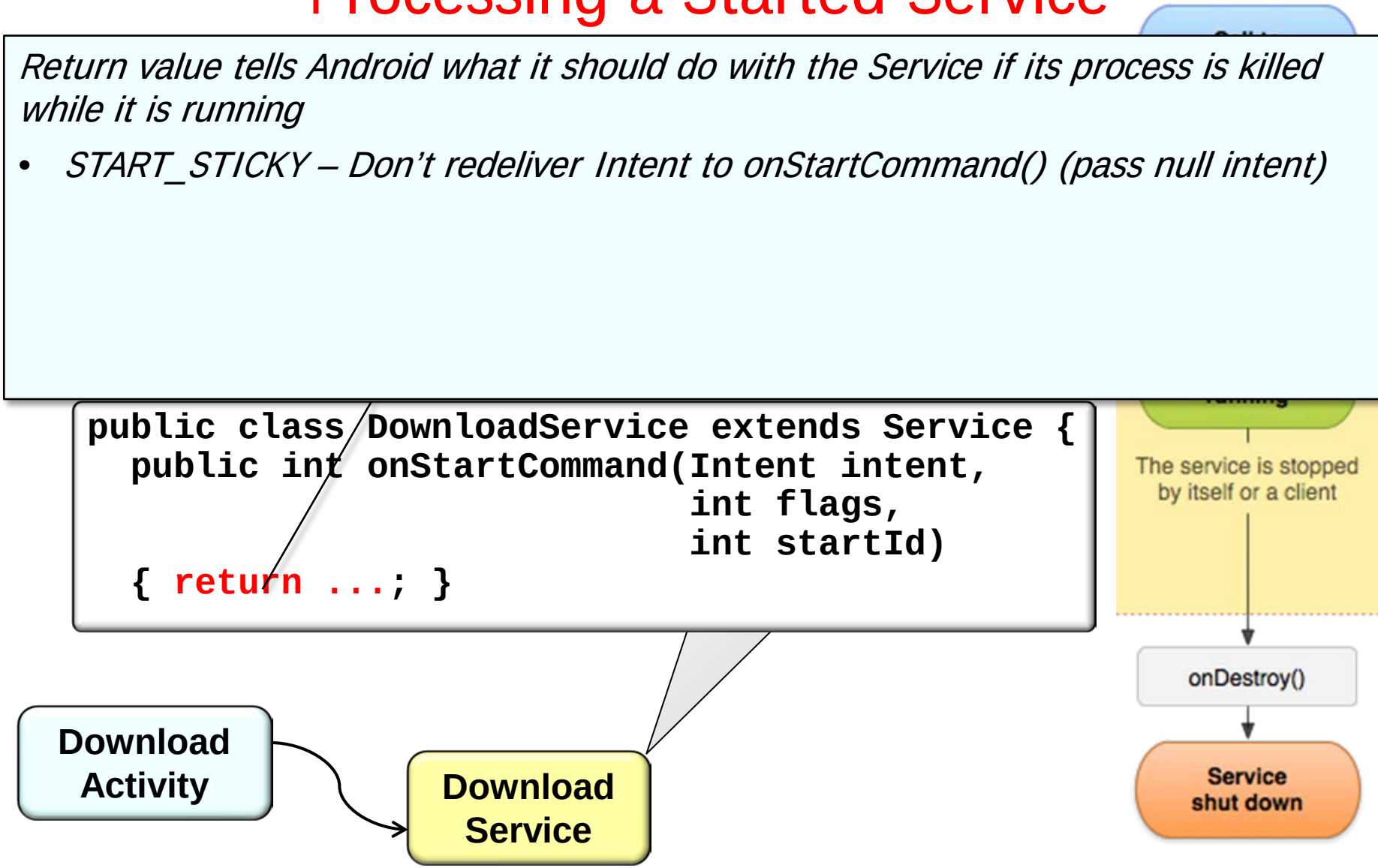


Processing a Started Service

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 ...; }
```

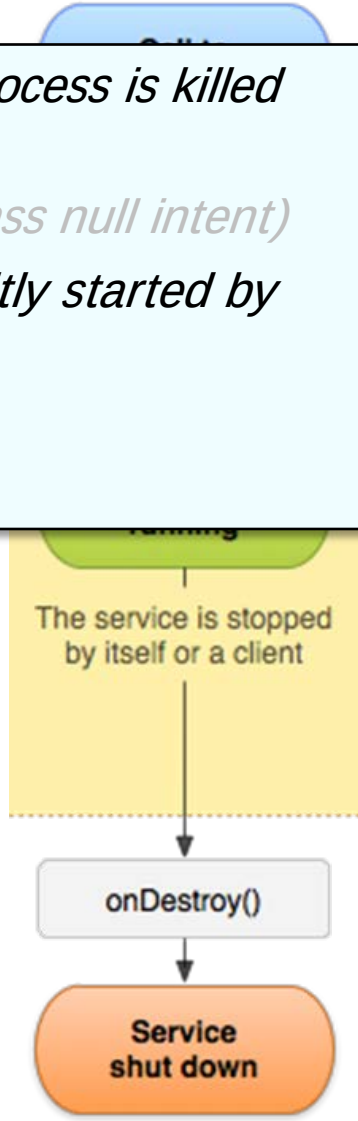


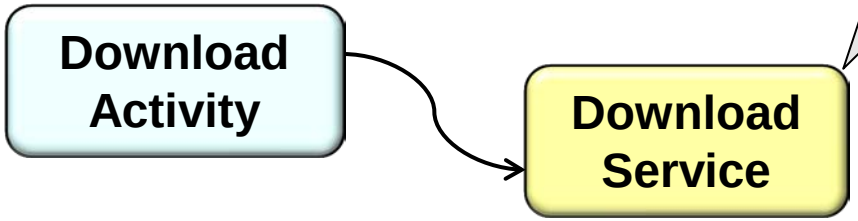
Processing a Started Service

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 ...; }
```



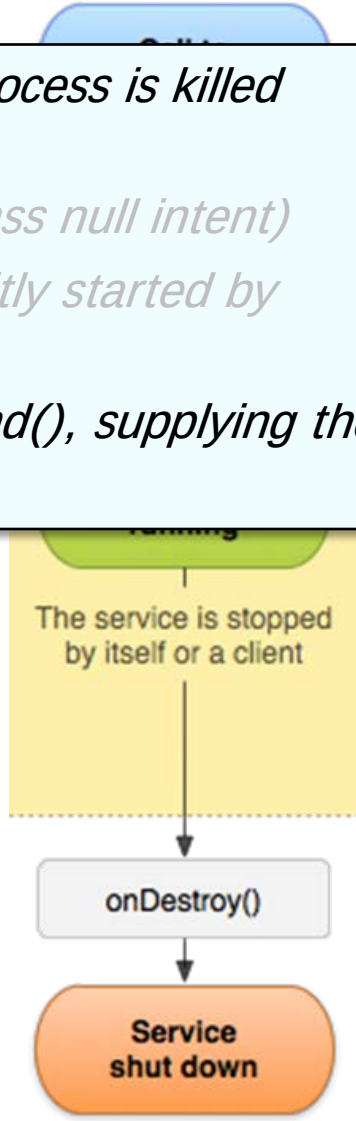


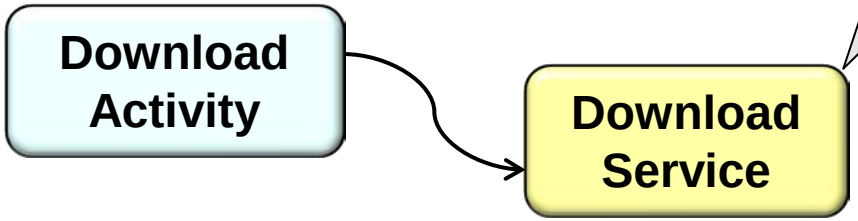
Processing a Started Service

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*

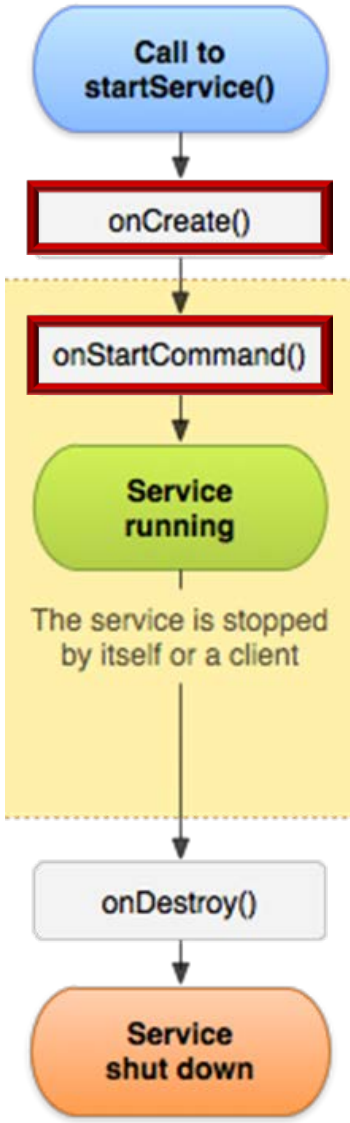
```
public class DownloadService extends Service {
  public int onStartCommand(Intent intent,
                            int flags,
                            int startId)
  { return ...; }
```



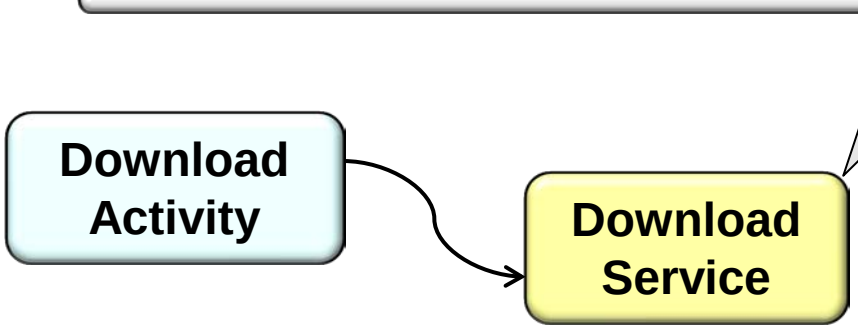


Processing a Started Service

- In response to startService(), Android will launch the Service if it's not already running
- Android then invokes the Service's onCreate() & onStartCommand() hook methods



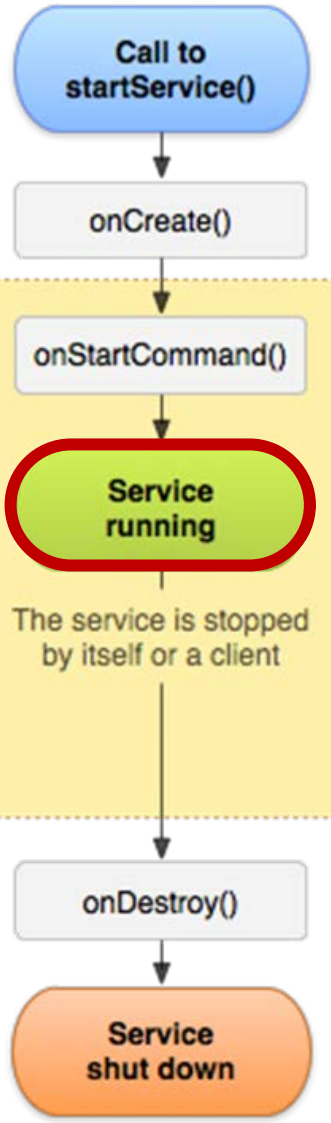

```
public class DownloadService extends Service {
  public int onStartCommand(Intent intent,
                            int flags,
                            int startId)
  { return ...; }
```



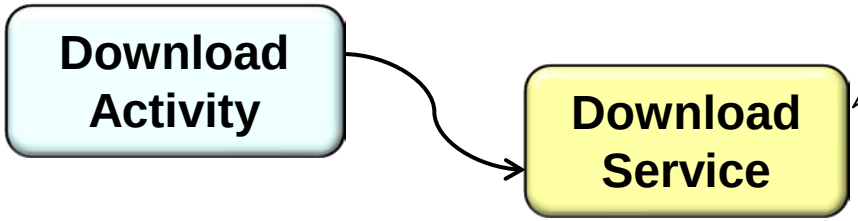
android-developers.blogspot.com.au/2010/02/service-api-changes-starting-with.html

Processing a Started Service

- In response to startService(), Android will launch the Service if it's not already running
- Android then invokes the Service's onCreate() & onStartCommand() hook methods
- A started service typically performs a single operation & often doesn't return a result to the client

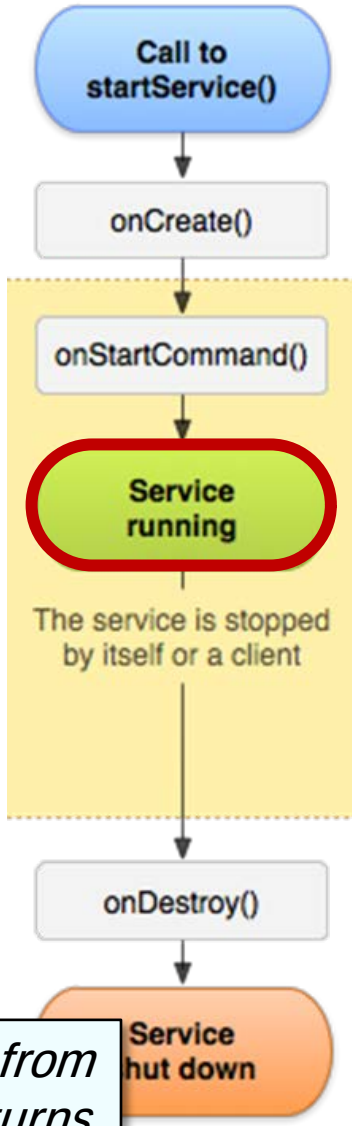



```
public class DownloadService extends Service {
  ...
  public void handleMessage(Message msg) {
      downloadImage((Intent) msg.obj);
      ...
```



Processing a Started Service

- In response to startService(), Android will launch the Service if it's not already running
- Android then invokes the Service's onCreate() & onStartCommand() hook methods
- 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);
      ...
```

Download Activity

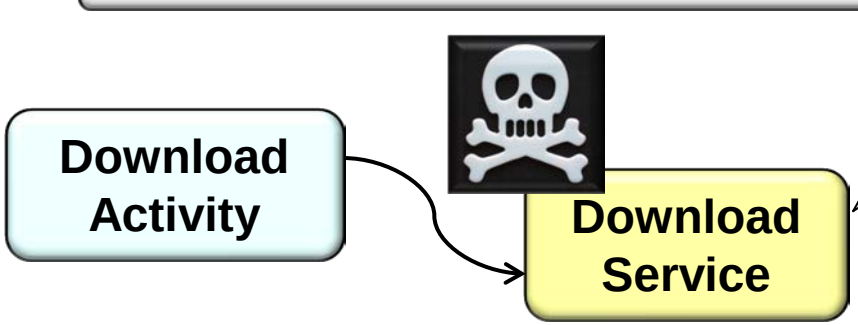
Download Service

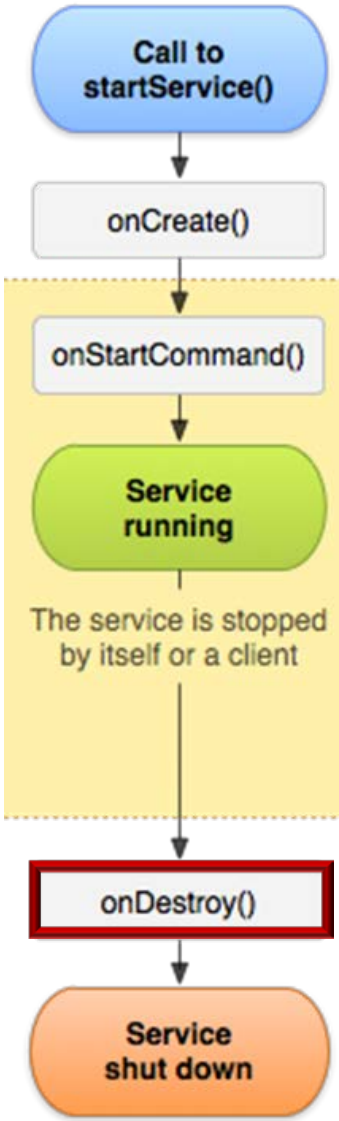
Retrieves an image from remote server & returns pathname to client

Stopping a Started Service

- When the operation is done, the Service can be stopped

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





Stopping a Started Service

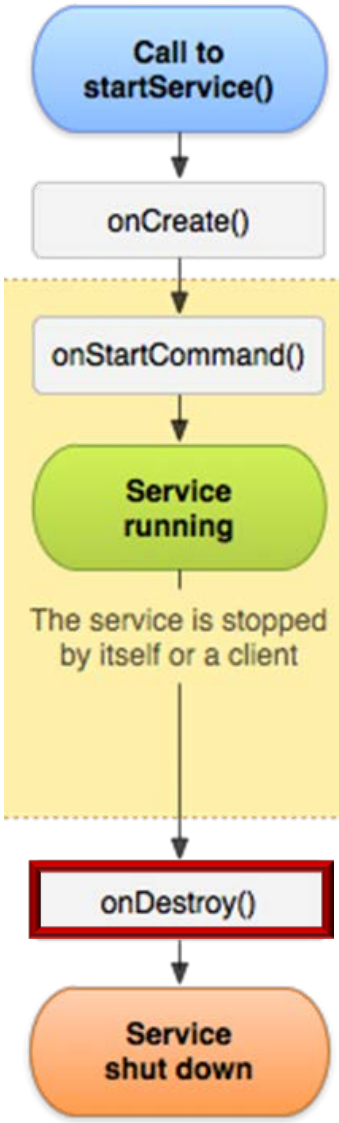
- When the operation is done, the Service can be stopped
  - e.g., the DownloadService call stopSelf() to shut itself down when it's done retrieving & processing an image

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

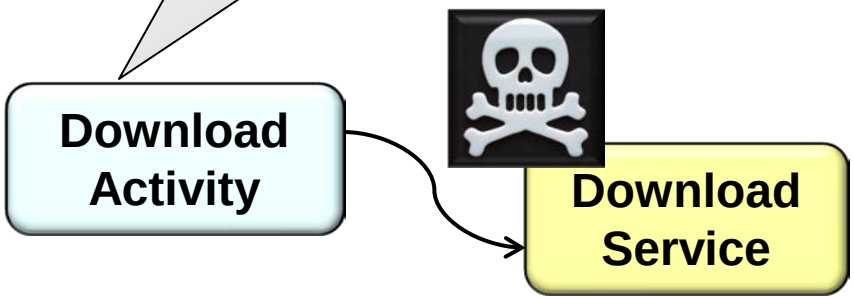
Download Service

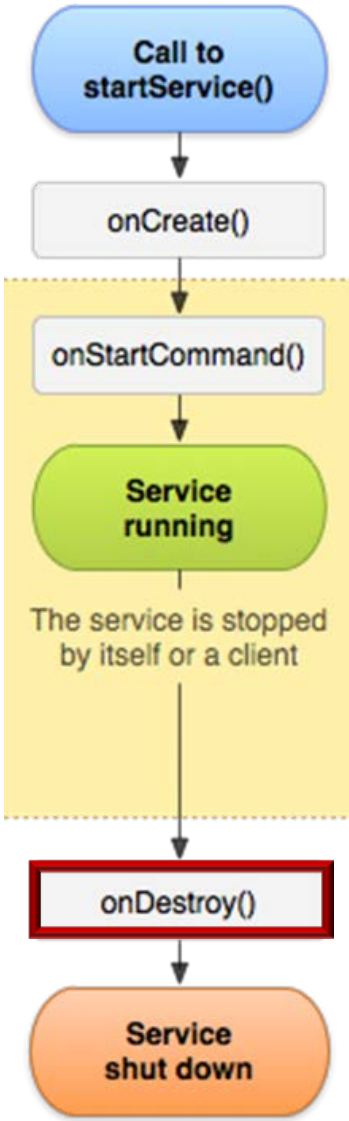


Stopping a Started Service

- When the operation is done, the Service can be stopped
  - e.g., the DownloadService call stopSelf() to shut itself down when it's done
  - Conversely, a Service can be shut down if stopService() is called by another component

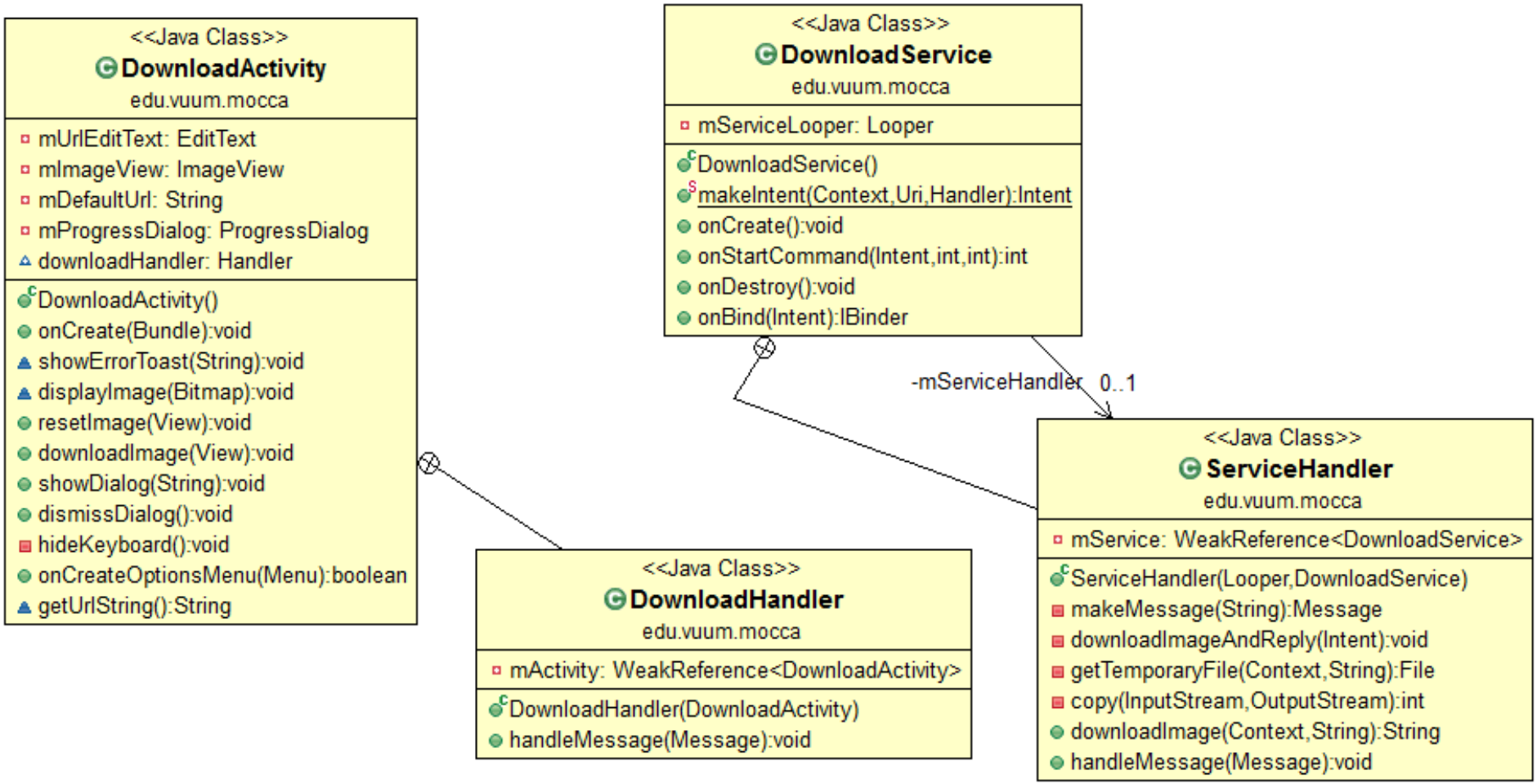
```
public class DownloadActivity ... {
  ...
  stopService(intent);
  ...
```



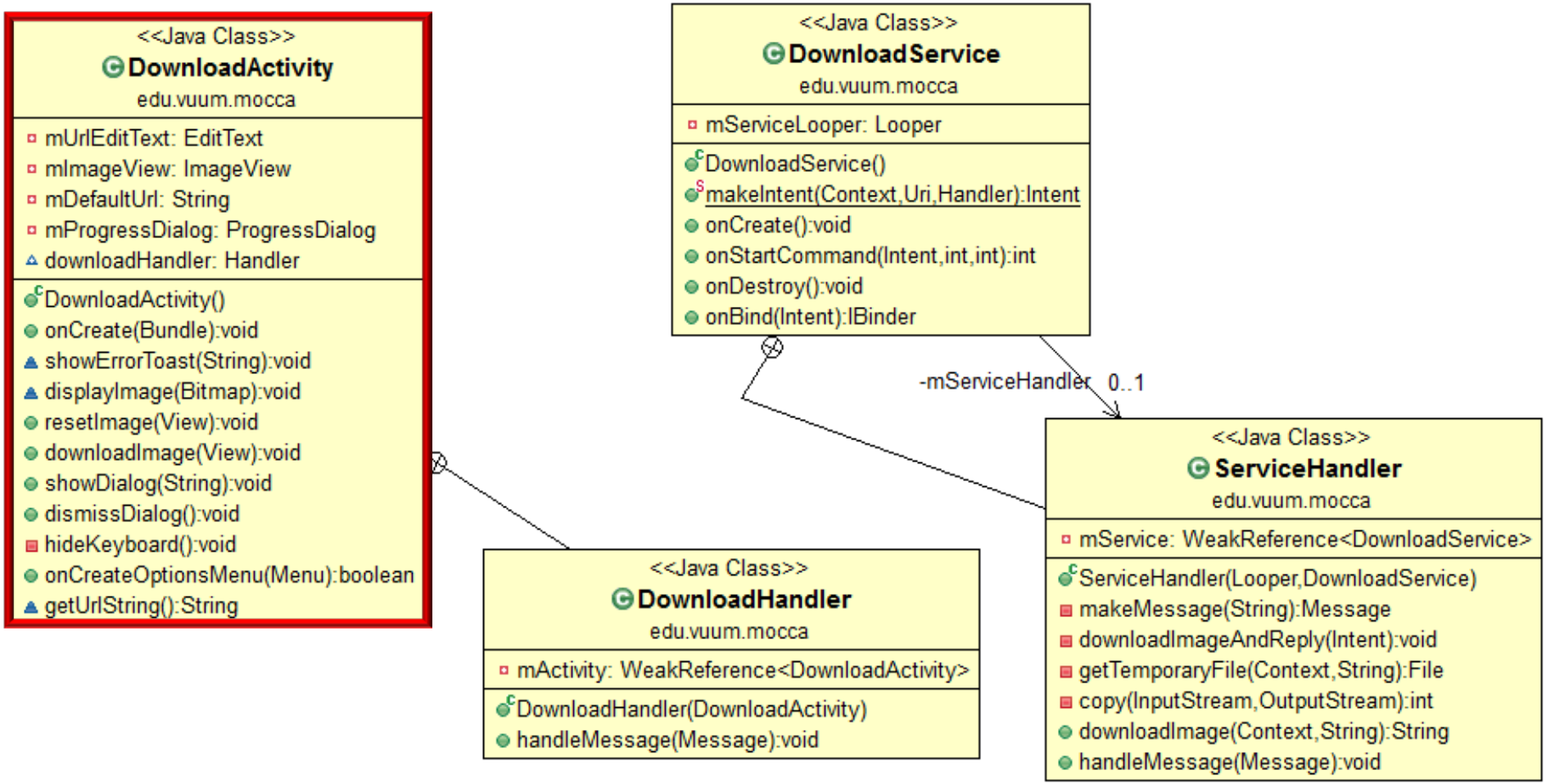


Design of the Download Application

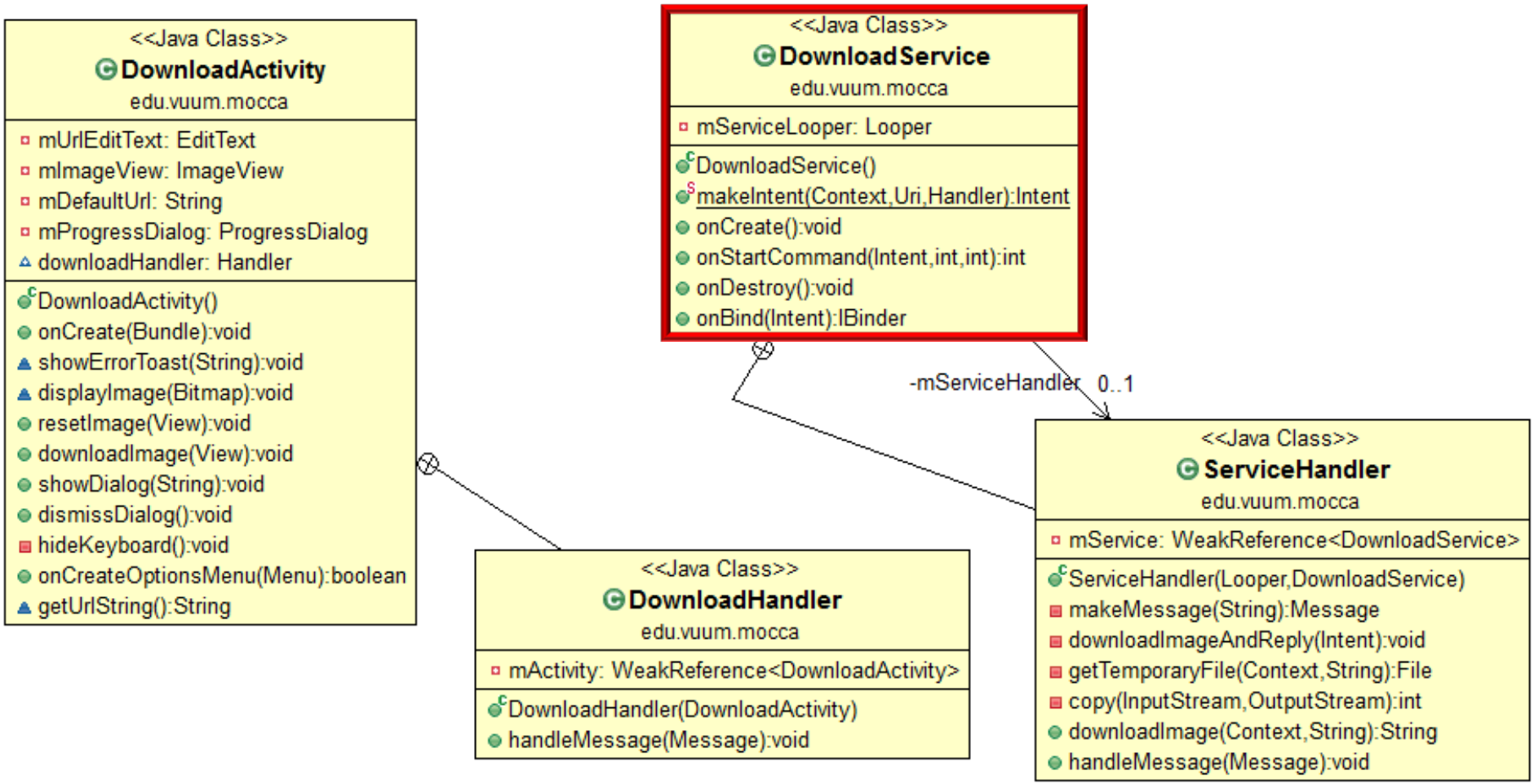
Design of the Download Application



Design of the Download Application

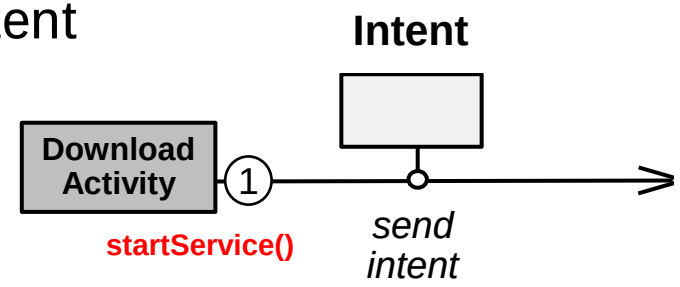


Design of the Download Application



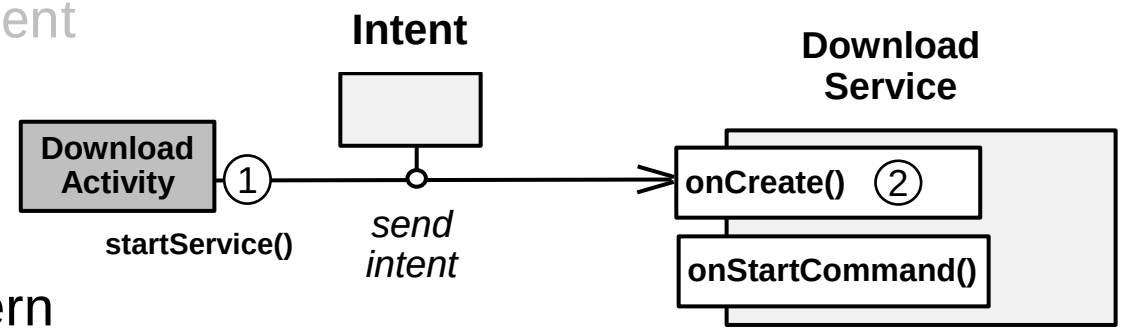
Design of the Download Application

- DownloadActivity sends an Intent via a call to startService()



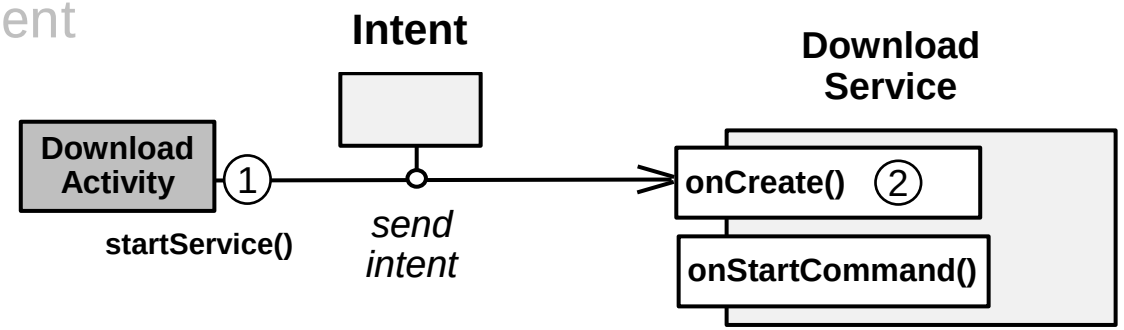
Design of the Download Application

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



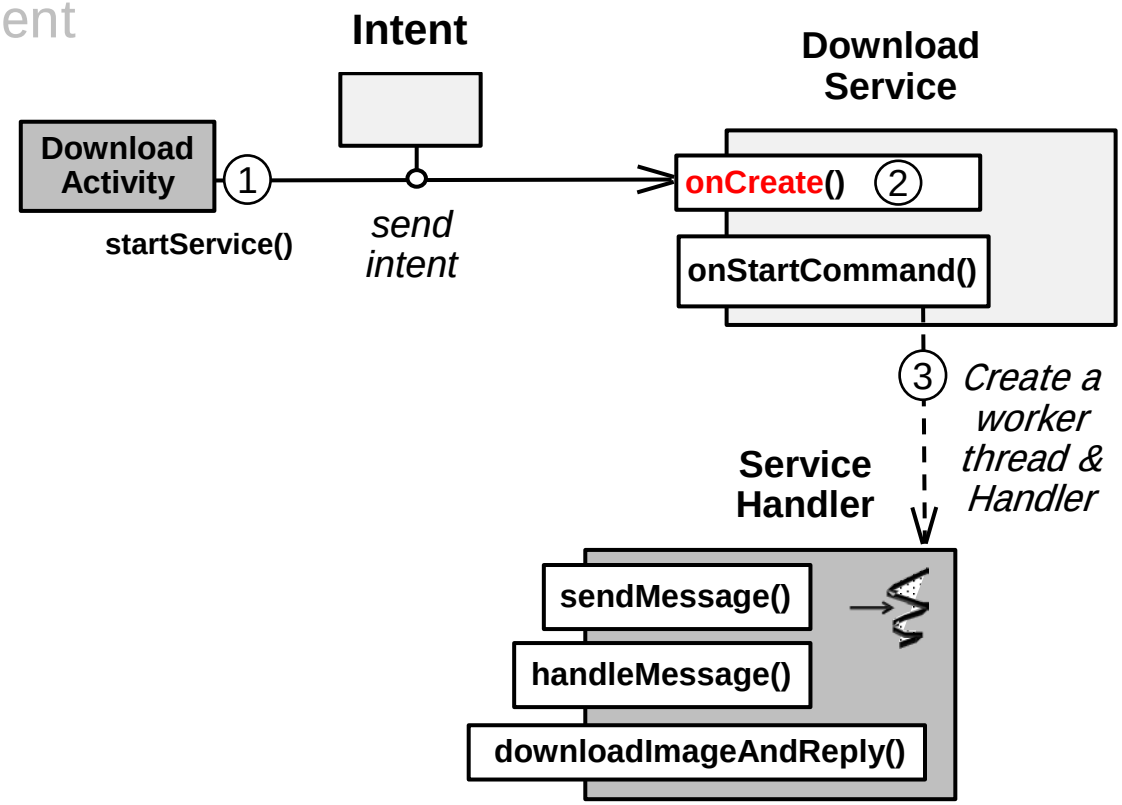
Design of the Download Application

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



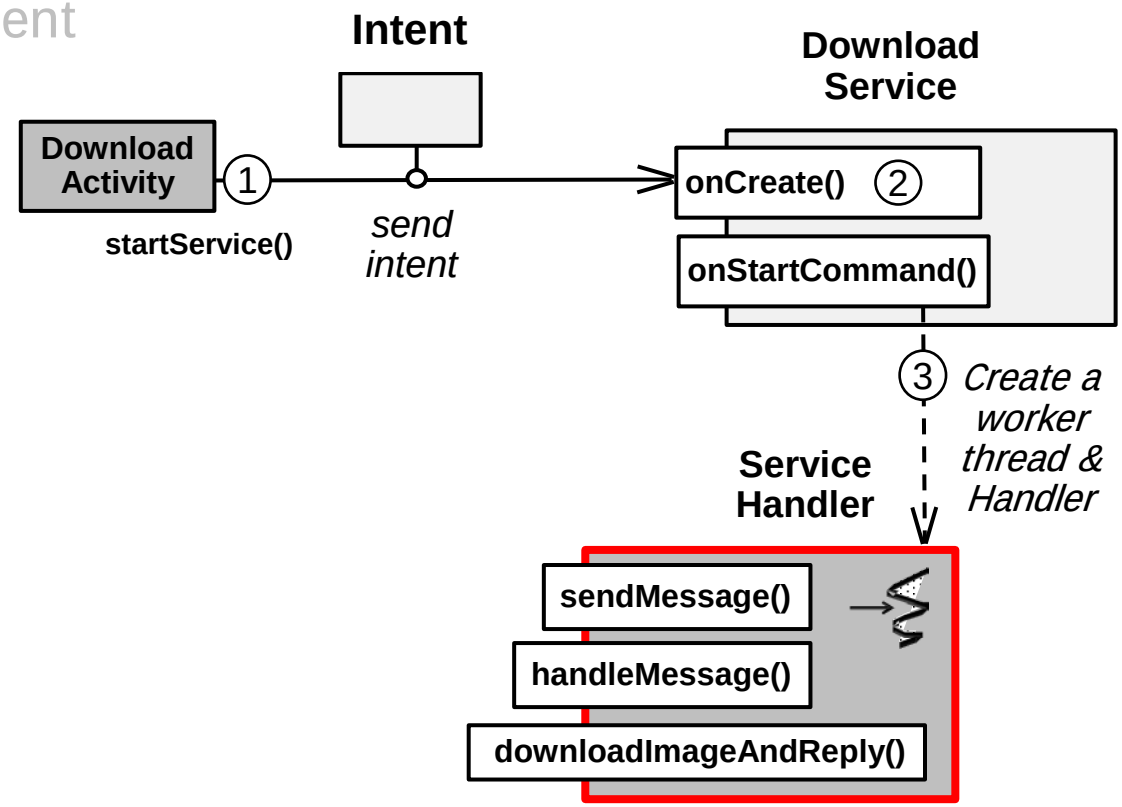
Design of the Download Application

- DownloadActivity sends an Intent via a call to startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
  1. Creates a ServiceHandler
     - Associated with a single HandlerThread



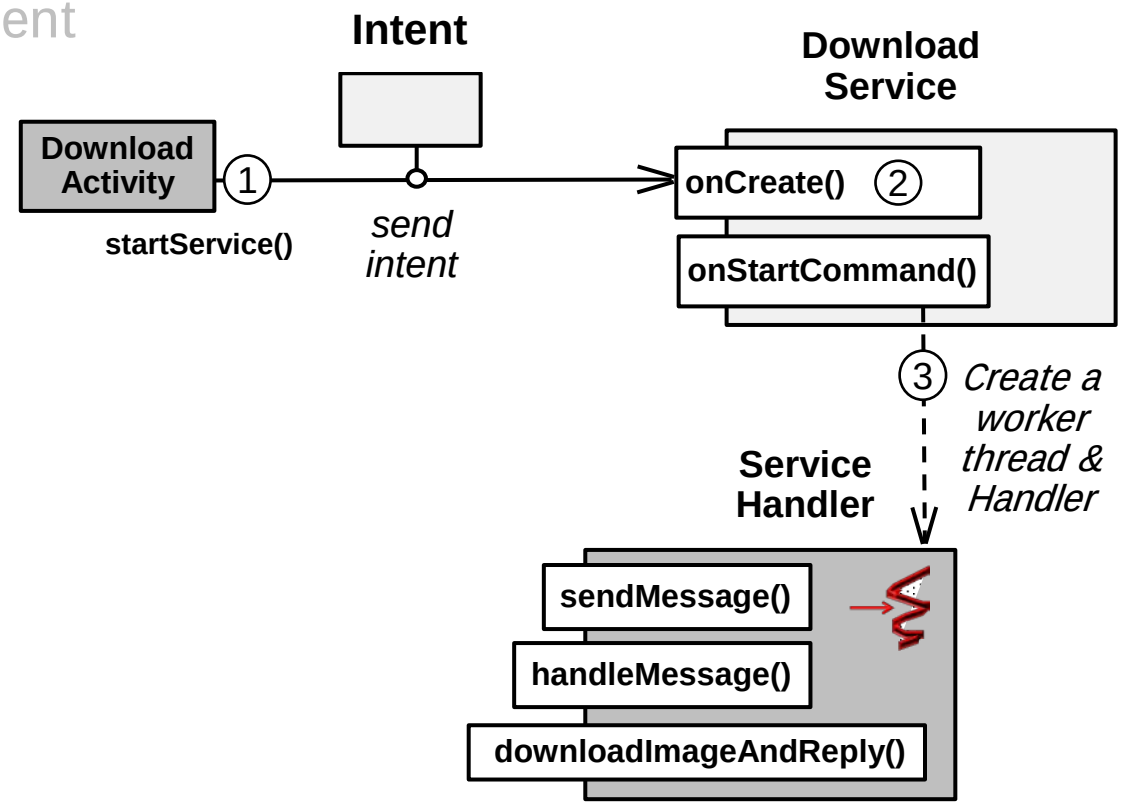
Design of the Download Application

- DownloadActivity sends an Intent via a call to startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
  1. Creates a ServiceHandler
     - Associated with a single HandlerThread



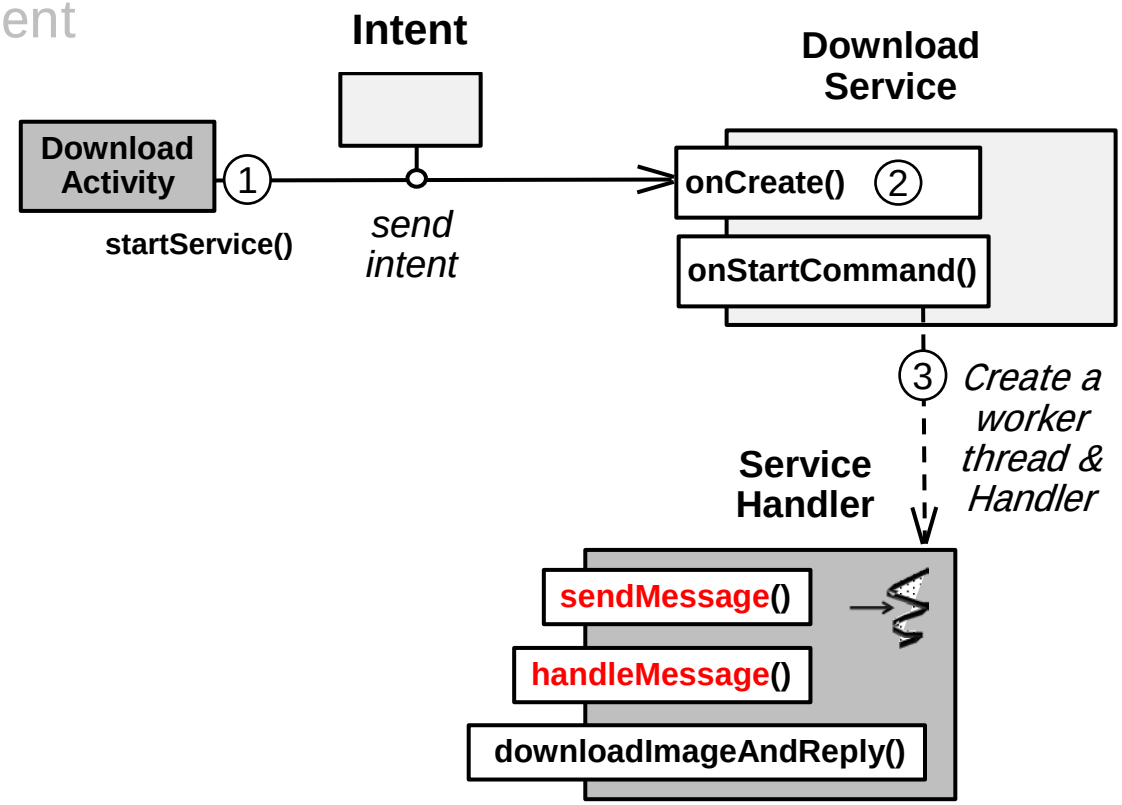
Design of the Download Application

- DownloadActivity sends an Intent via a call to startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
  1. Creates a ServiceHandler
     - Associated with a single HandlerThread



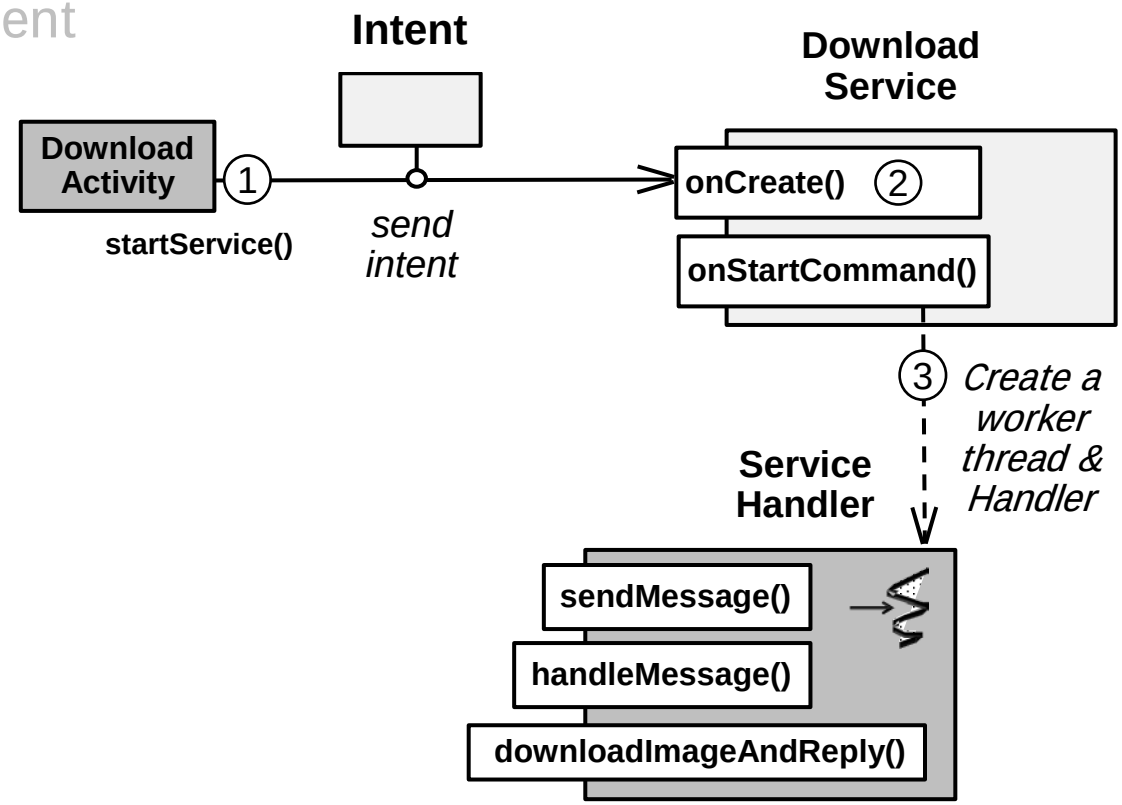
Design of the Download Application

- DownloadActivity sends an Intent via a call to 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

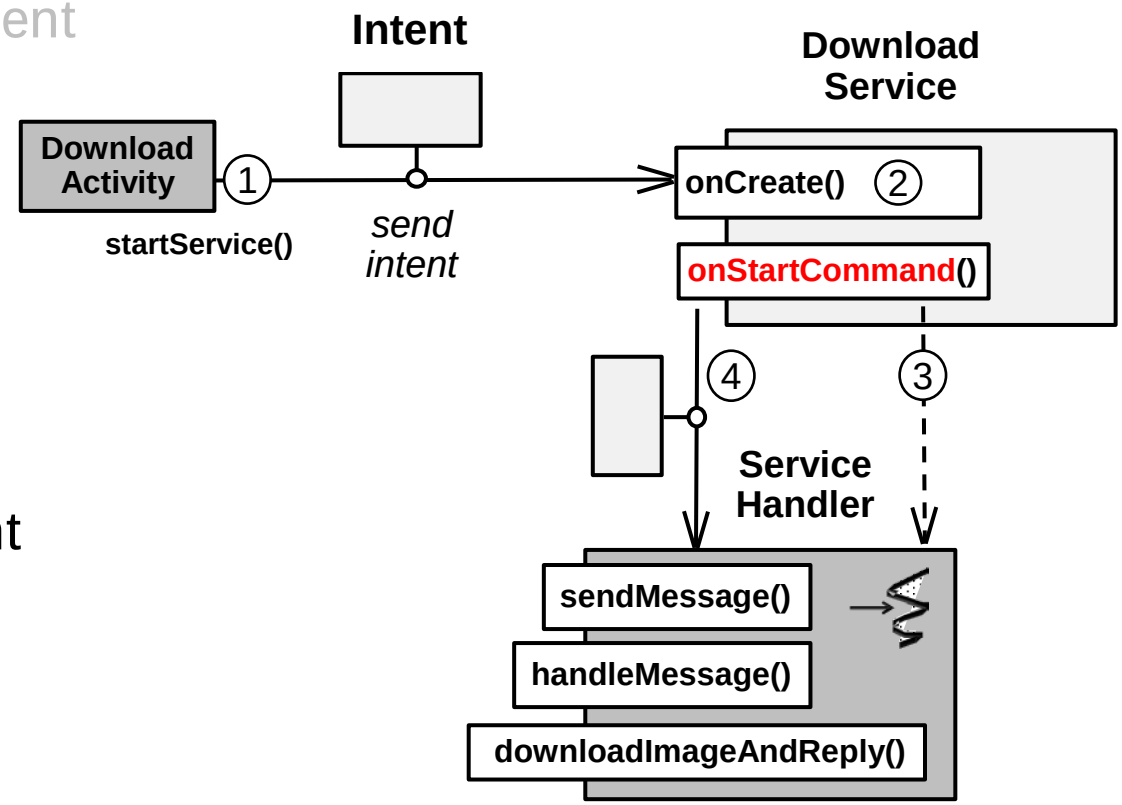
- DownloadActivity sends an Intent via a call to startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
  1. Creates a ServiceHandler
     - Associated with a single HandlerThread



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

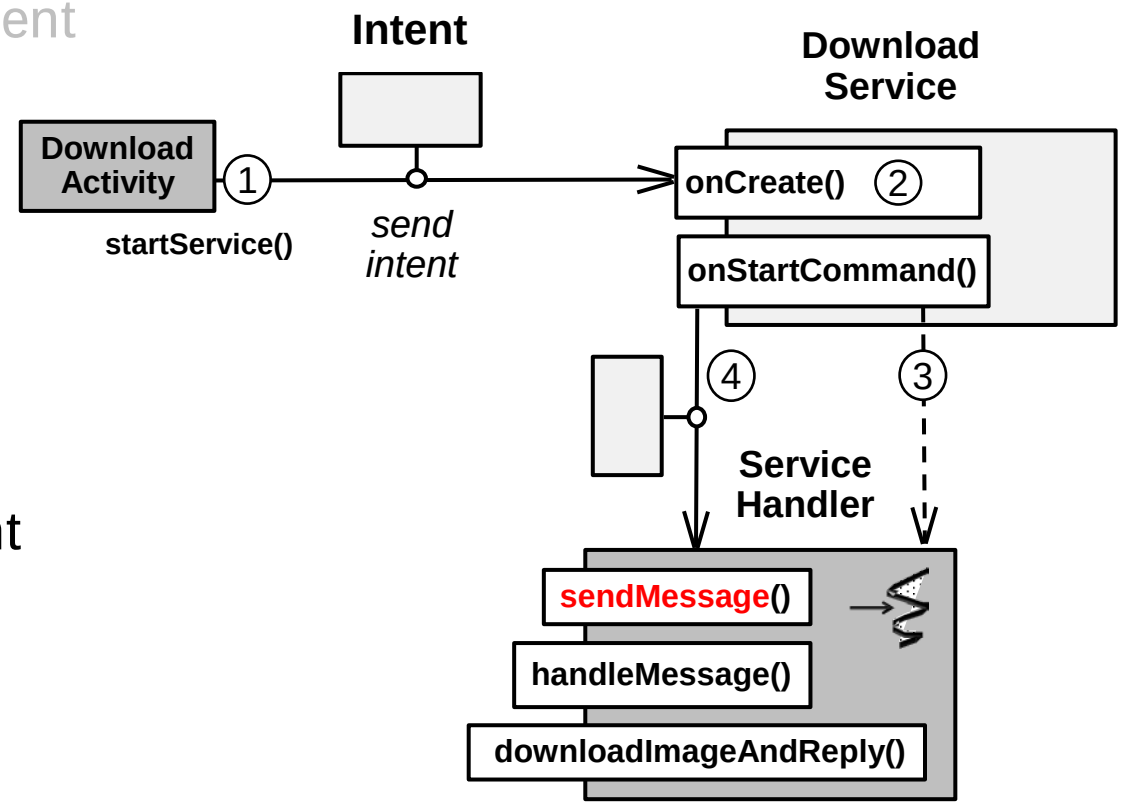
Design of the Download Application

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



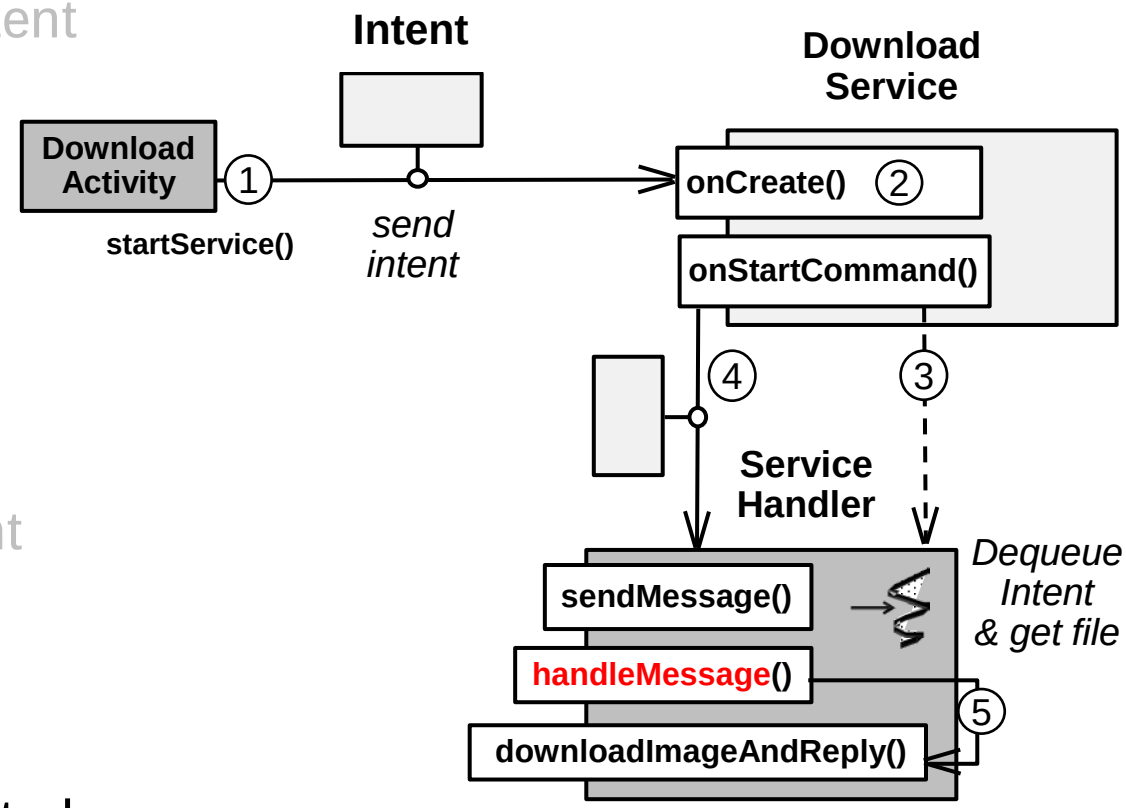
Design of the Download Application

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



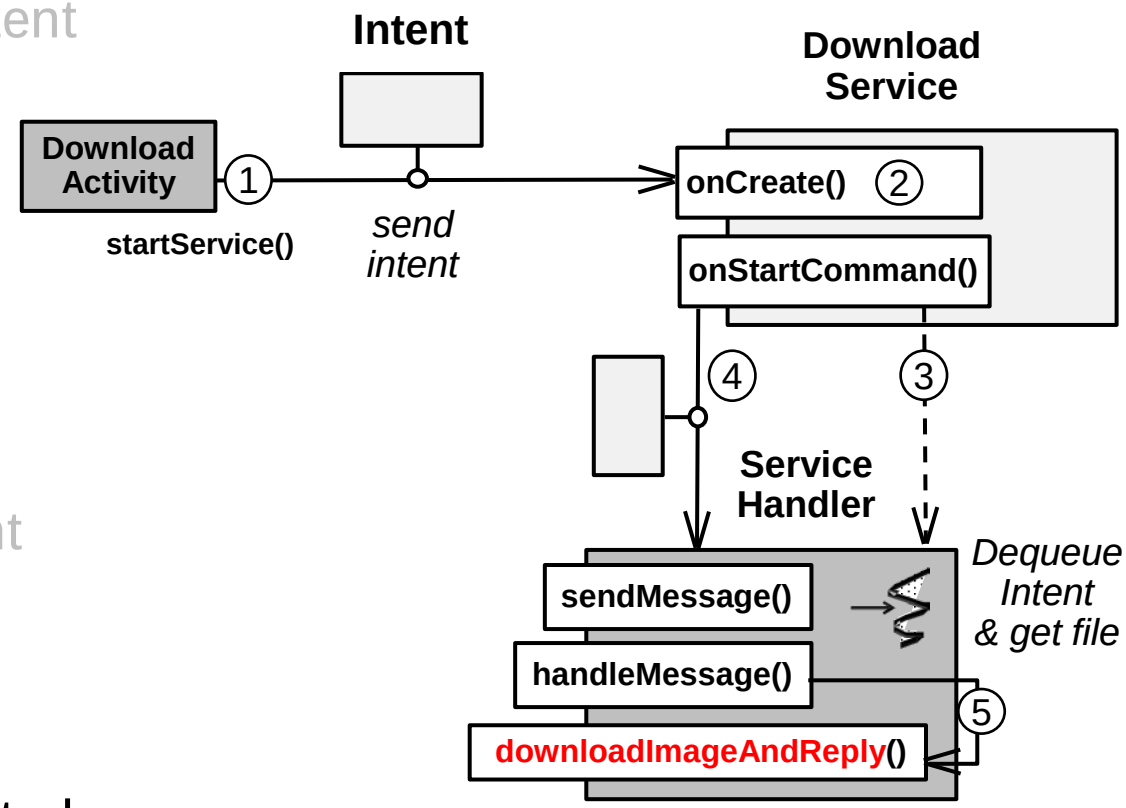
Design of the Download Application

- DownloadActivity sends an Intent via a call to 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



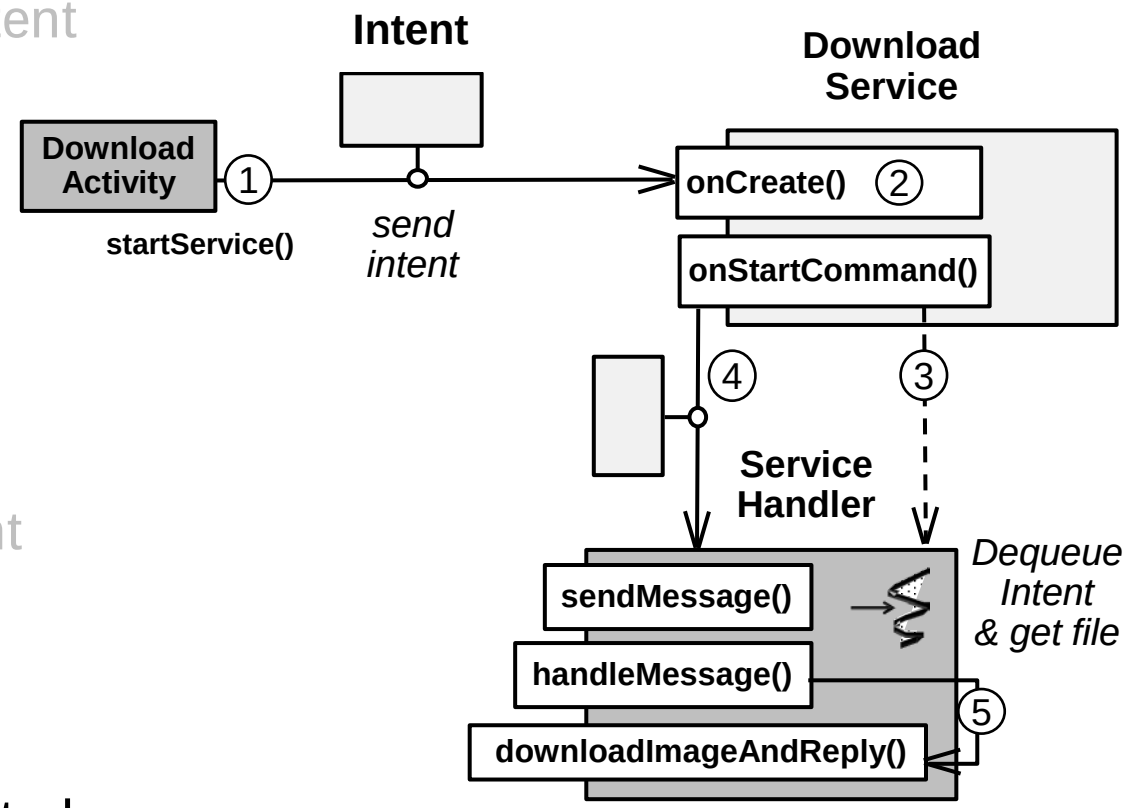
Design of the Download Application

- DownloadActivity sends an Intent via a call to 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

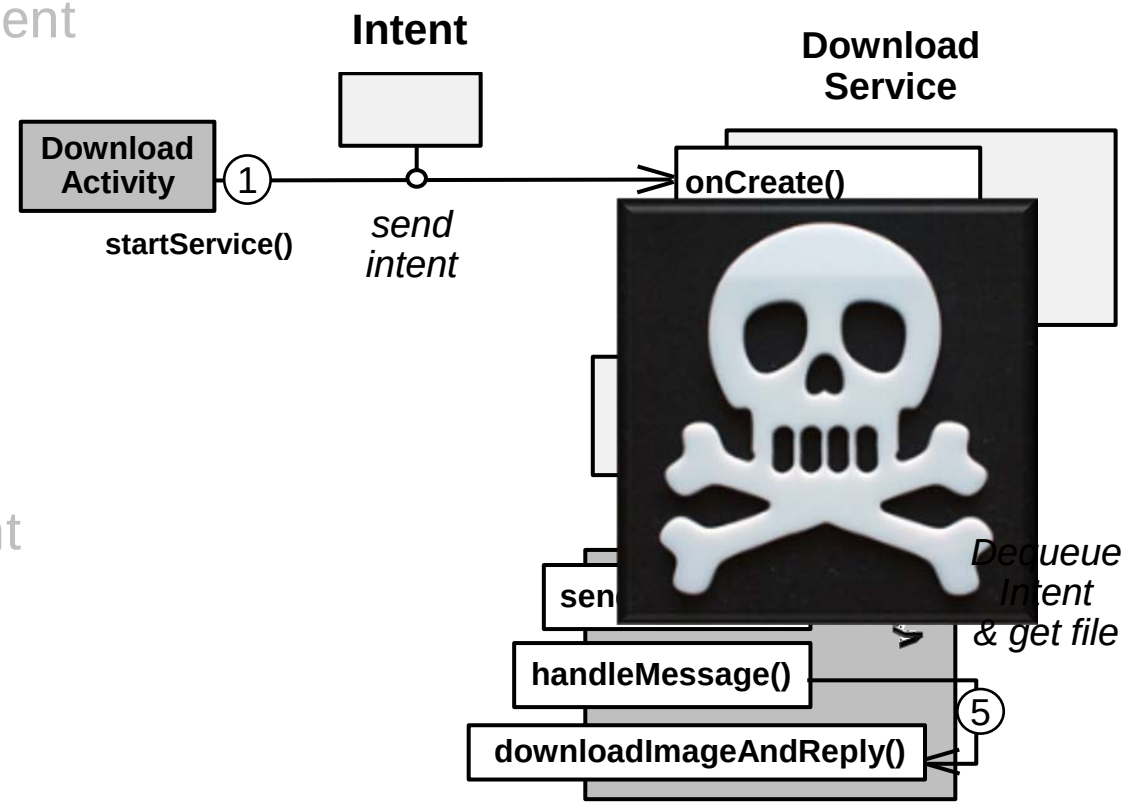
- DownloadActivity sends an Intent via a call to startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
  1. Creates a ServiceHandler
  2. Receives & queues an Intent in the ServiceHandler
  3. ServiceHandler processes Intent "in the background"
     -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



See upcoming parts on "Activity & Service Communication"

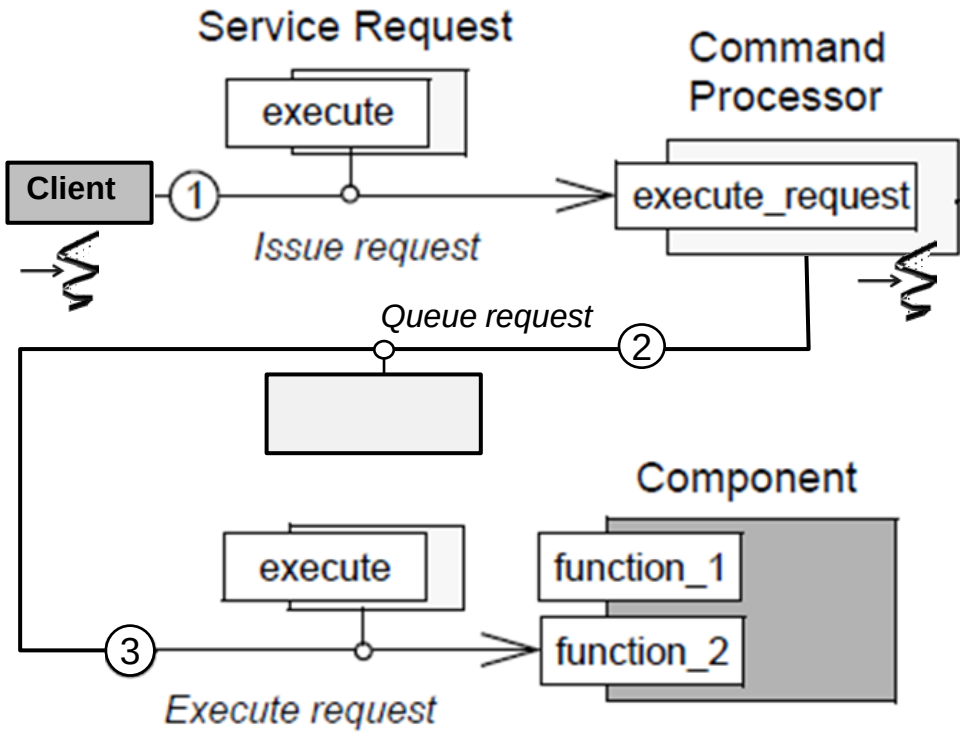
Design of the Download Application

- DownloadActivity sends an Intent via a call to 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

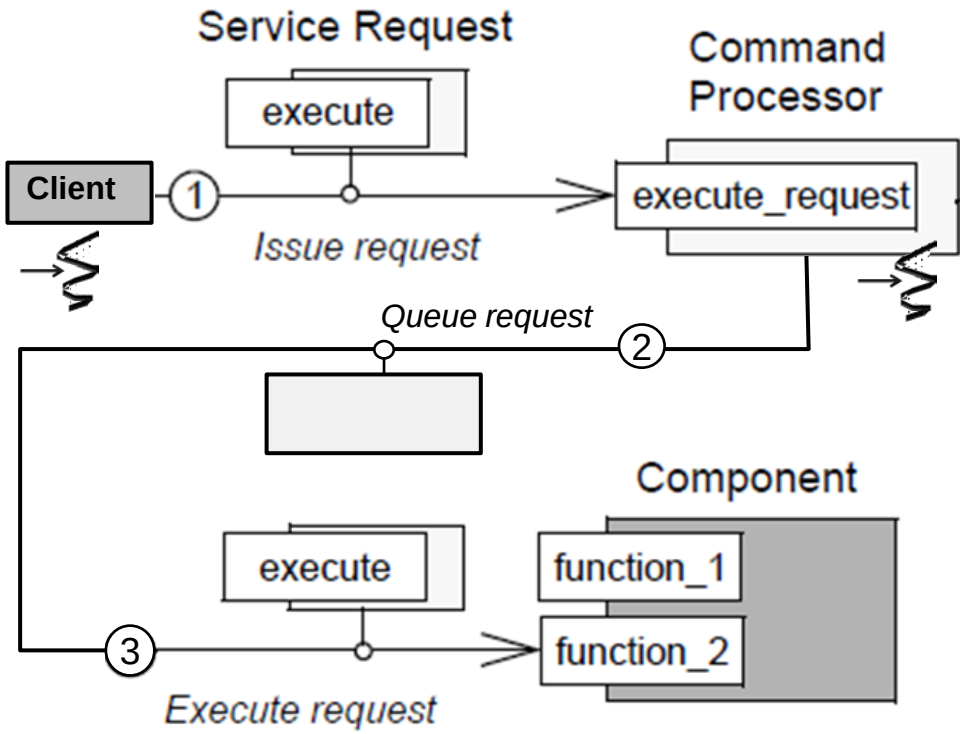
- DownloadActivity sends an Intent via a call to startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
- This design is guided by the *Command Processor* pattern



www.dre.vanderbilt.edu/~schmidt/PDF/CommandRevisited.pdf has more info

Design of the Download Application

- DownloadActivity sends an Intent via a call to startService()
- The DownloadService is launched on-demand
- DownloadService performs four main actions
- This design is guided by the *Command Processor* pattern
  - Offload tasks from application's main Thread to a single backgroundThread

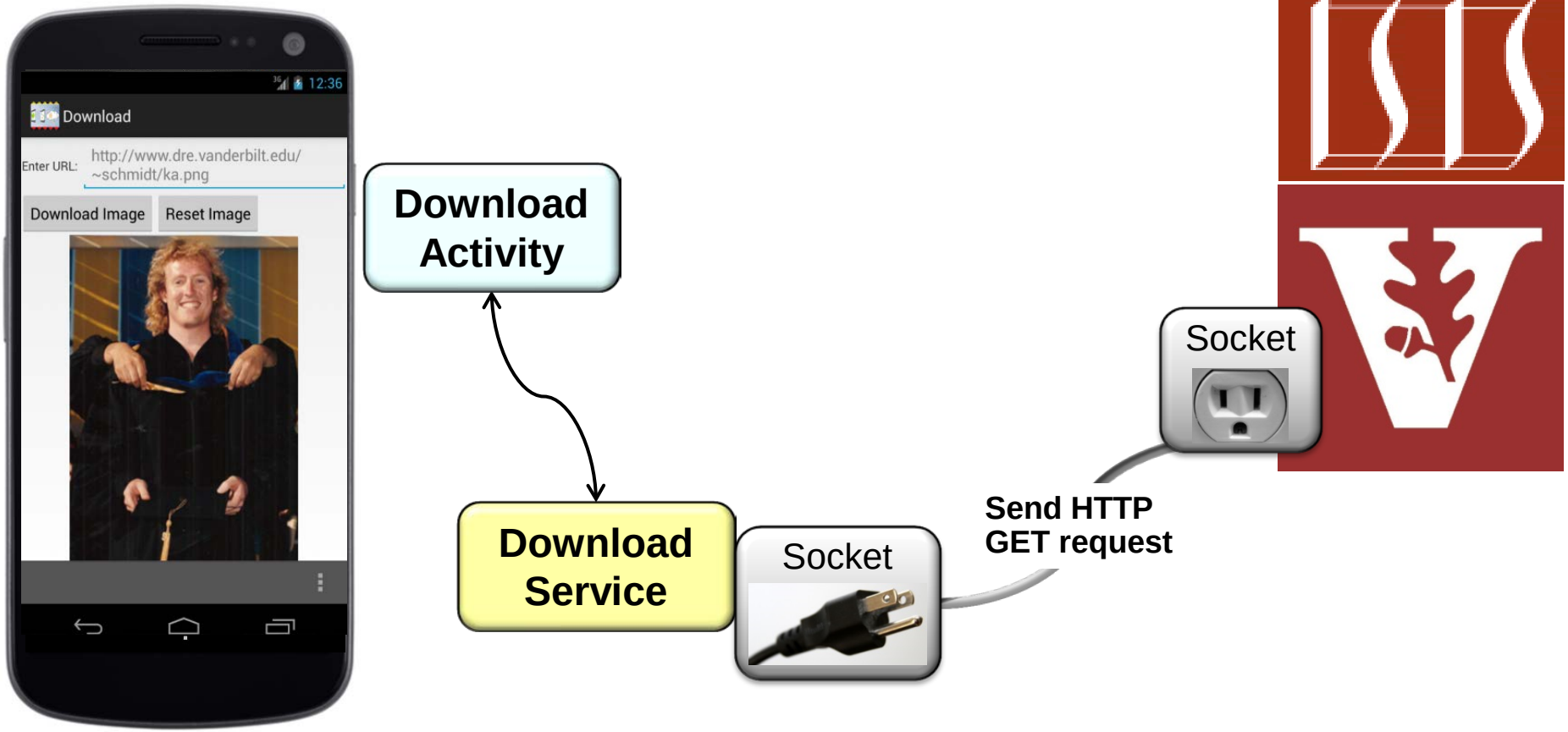


This pattern works if a Service needn't handle multiple requests concurrently

Summary

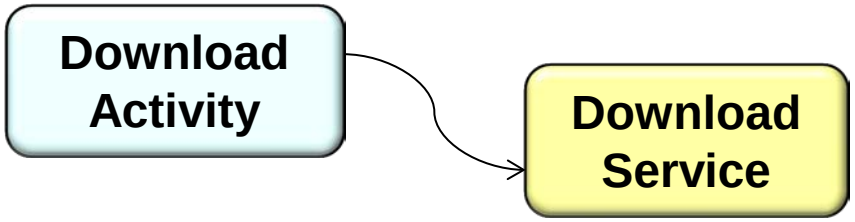
Summary

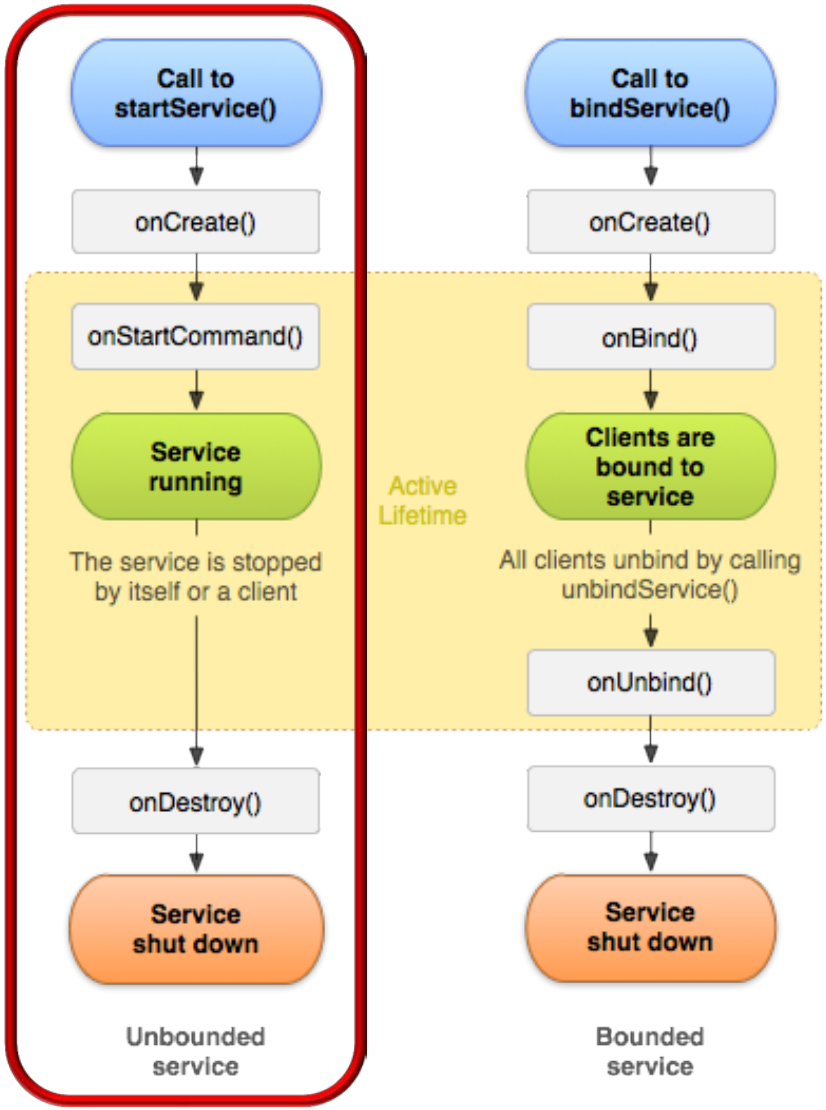
- The Download Application uses a Started Service to retrieve & display an image from a remote server



Summary

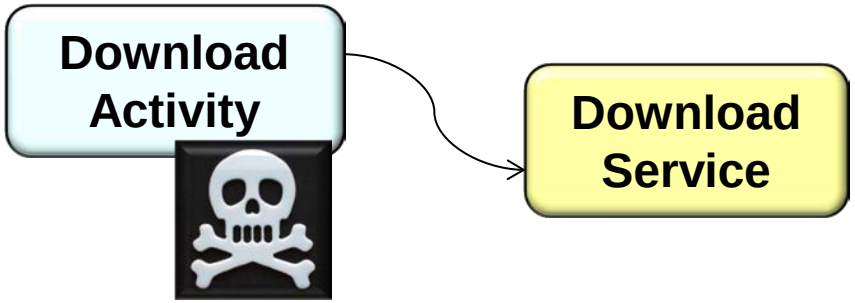
- The Download Application uses a Started Service to retrieve & display an image from a remote server
- When a Started Service is launched, it has a lifecycle that's independent of the component that started it

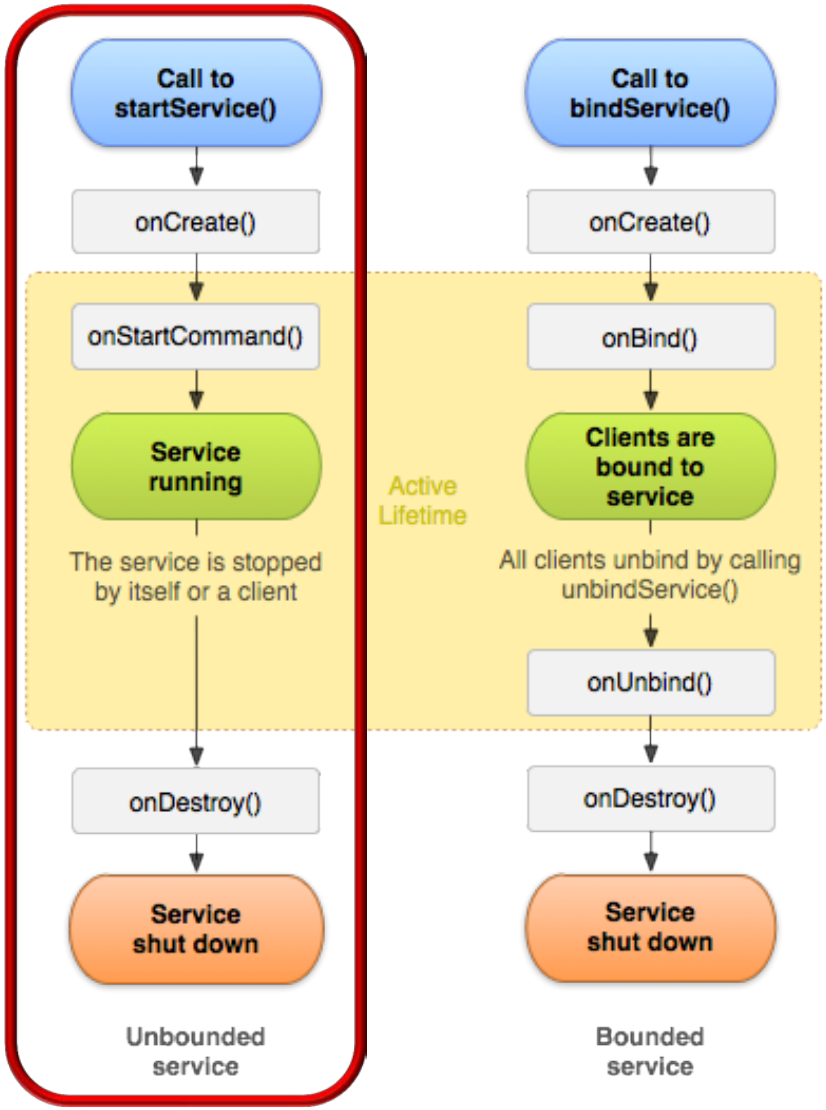




Summary

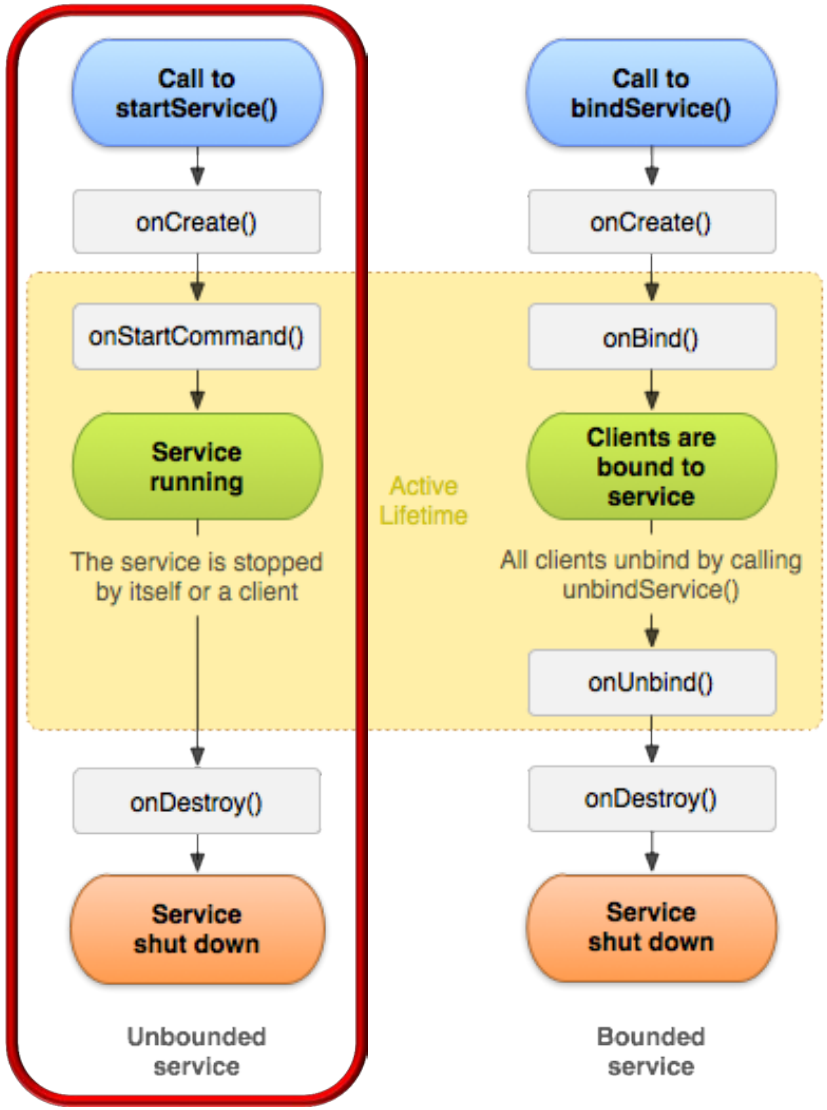
- The Download Application uses a Started Service to retrieve & display an image from a remote server
- When a Started Service is launched, it has a lifecycle that's independent of the component that started it
  - A Service can run in the background indefinitely, even if the component that started it is destroyed





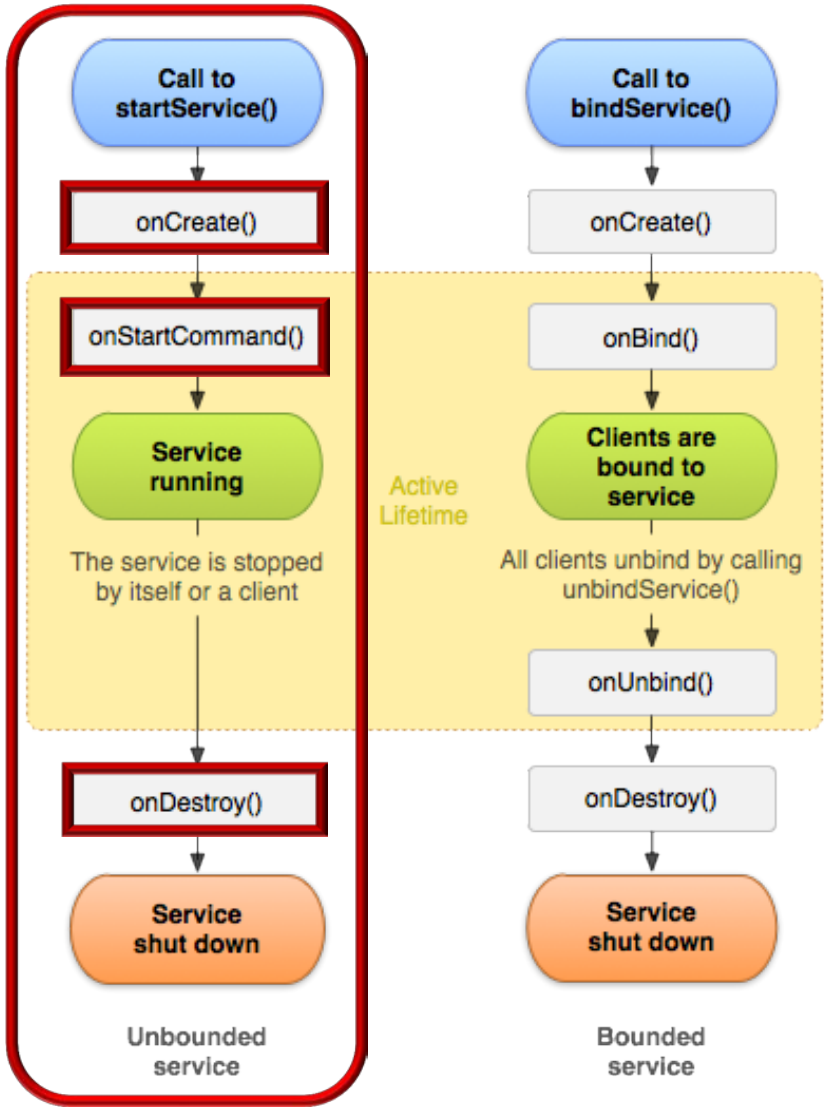
Summary

- The Download Application uses a Started Service to retrieve & display an image from a remote server
- When a Started Service is launched, it has a lifecycle that's independent of the component that started it
- Started Services are driven by inversion of control



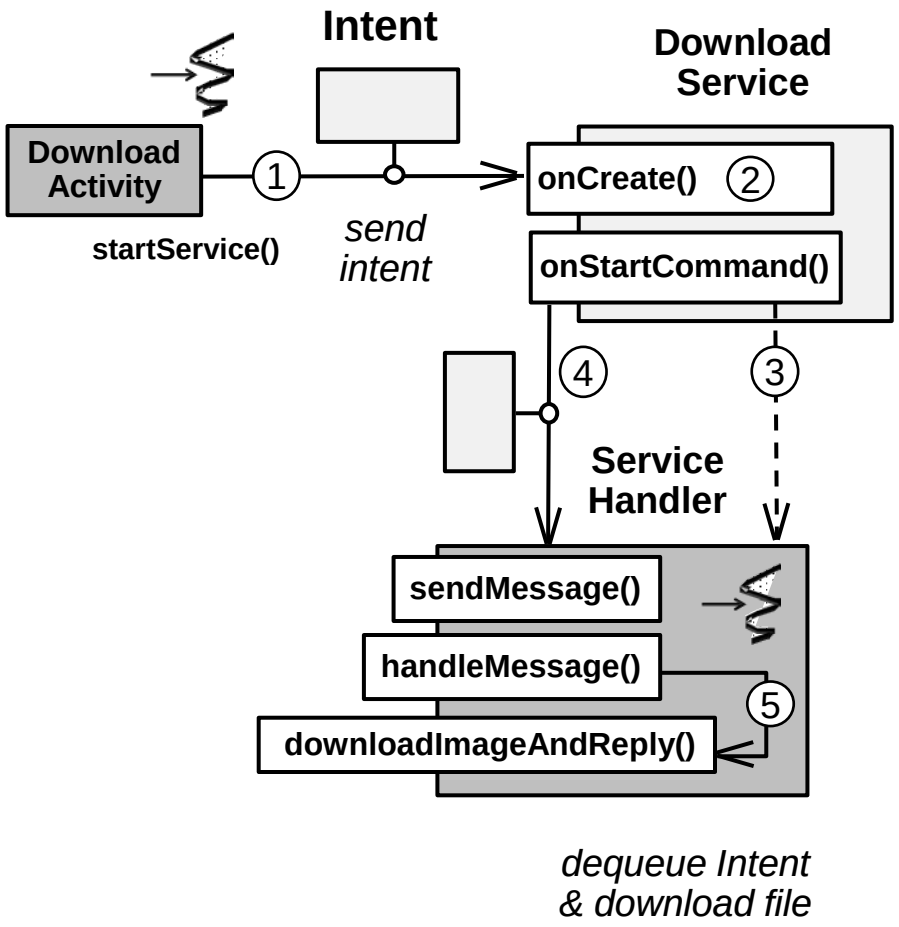
Summary

- The Download Application uses a Started Service to retrieve & display an image from a remote server
- When a Started Service is launched, it has a lifecycle that's independent of the component that started it
- Started Services are driven by inversion of control



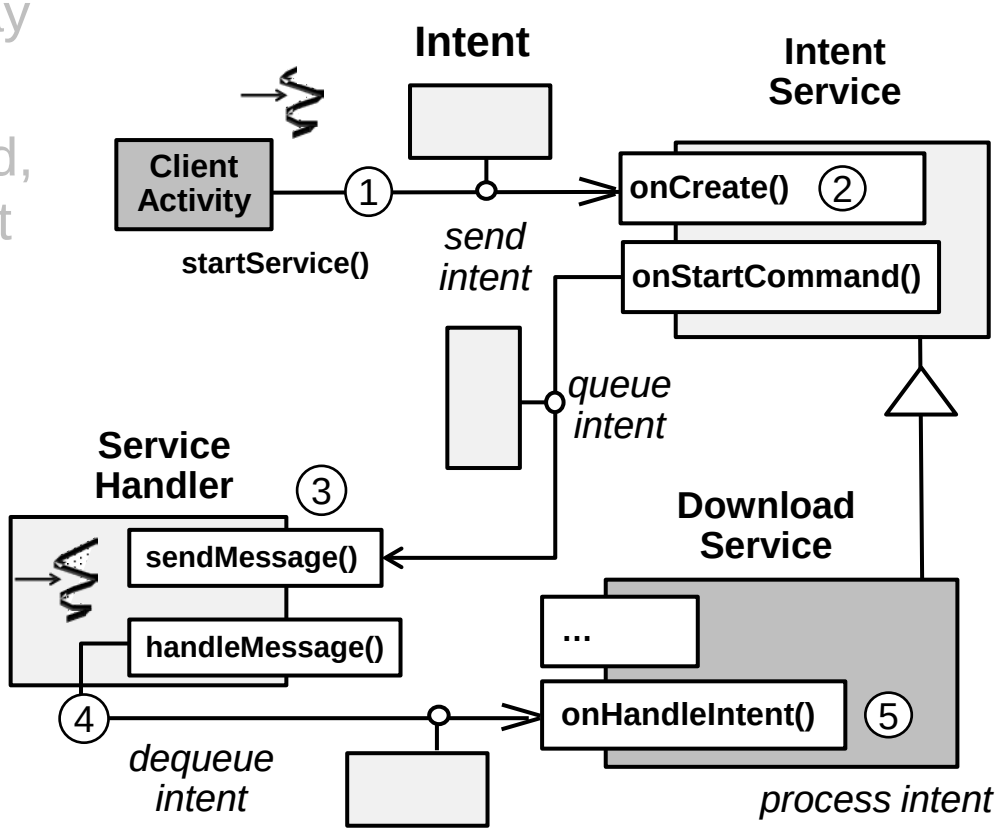
Summary

- The Download Application uses a Started Service to retrieve & display an image from a remote server
- When a Started Service is launched, it has a lifecycle that's independent of the component that started it
- Started Services are driven by inversion of control
- The Download Application implementation is guided by a common Android idiom for concurrent Service processing



Summary

- The Download Application uses a Started Service to retrieve & display an image from a remote server
- When a Started Service is launched, it has a lifecycle that's independent of the component that started it
- Started Services are driven by inversion of control
- The Download Application implementation is guided by a common Android idiom for concurrent Service processing
  - This idiom is based on the *Command Processor* pattern



See upcoming parts on “The Command Processor Pattern”