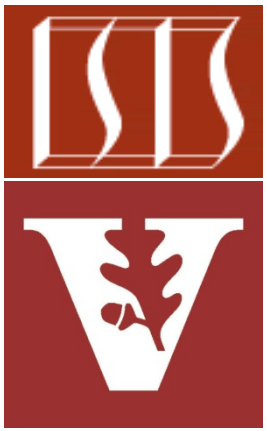
# Android Services & Security: Android IntentService

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



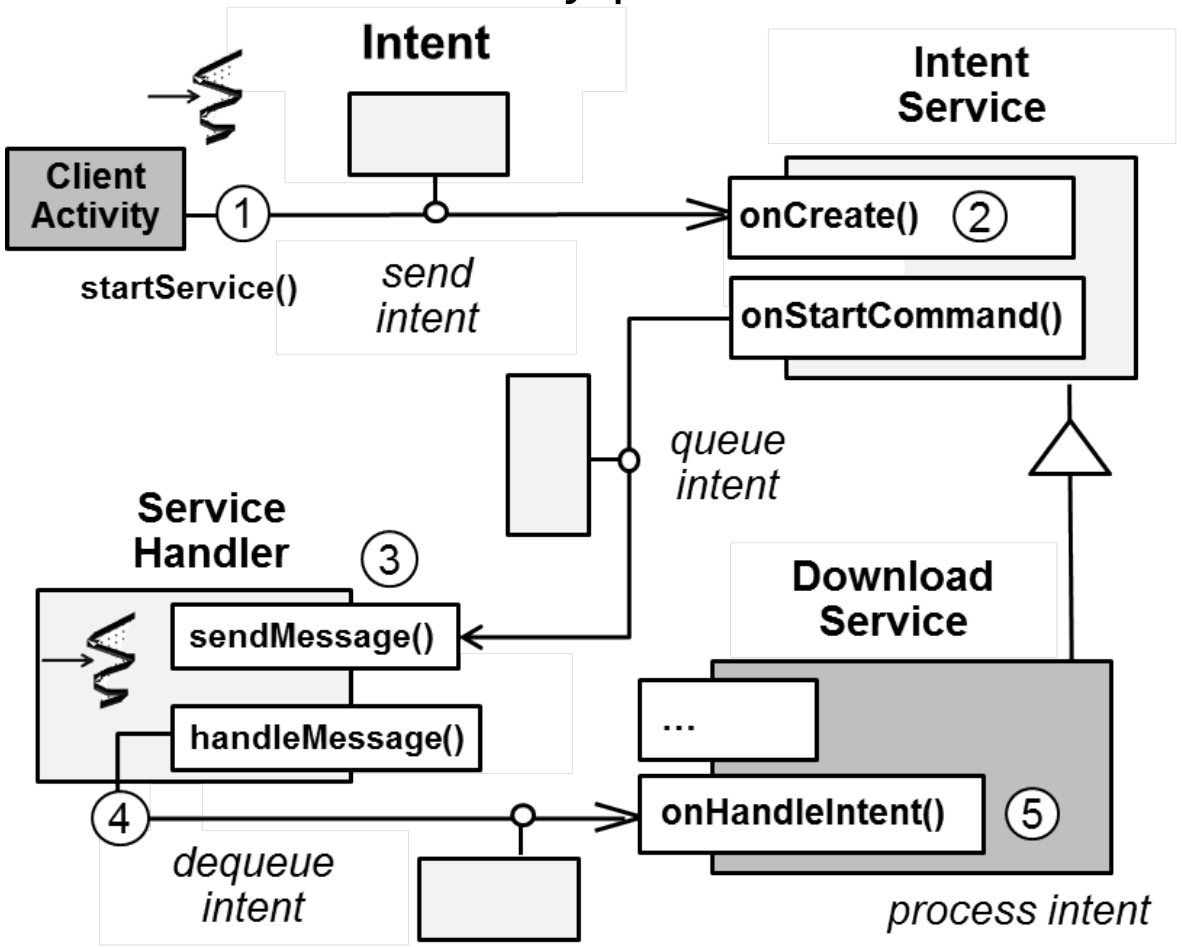
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

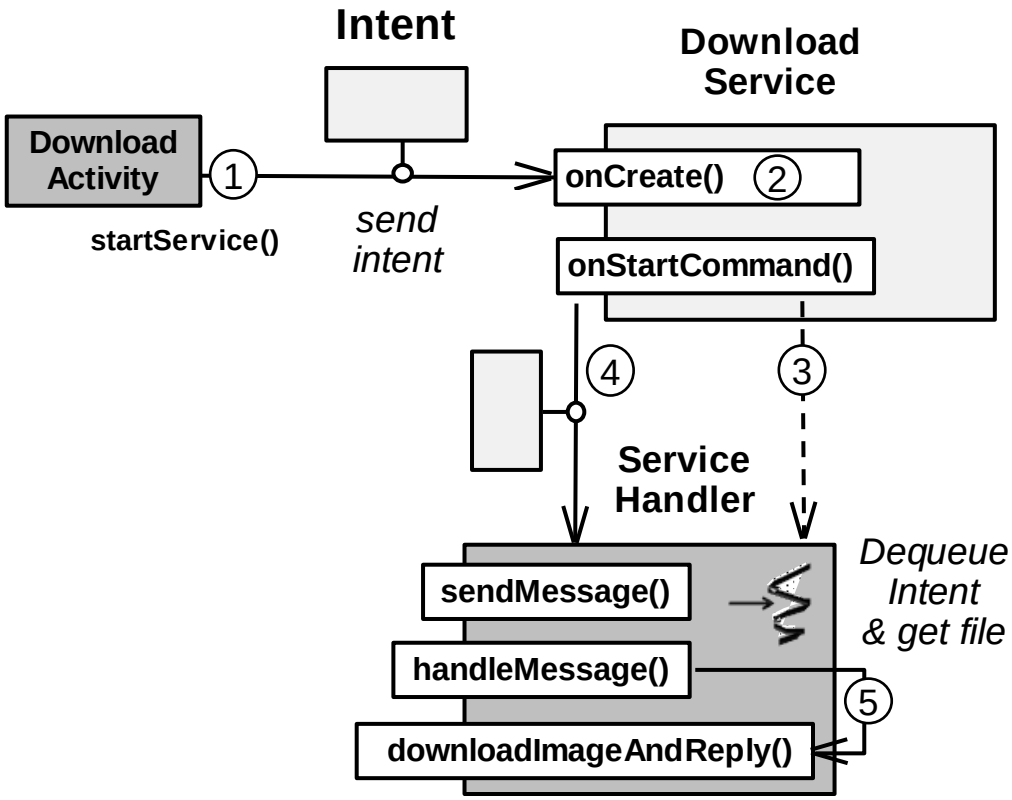
# Learning Objectives in this Part of the Module

- Understand how the Android IntentService provides a framework for programming Started Services that concurrently process commands expressed as Intents



# Motivation for IntentService

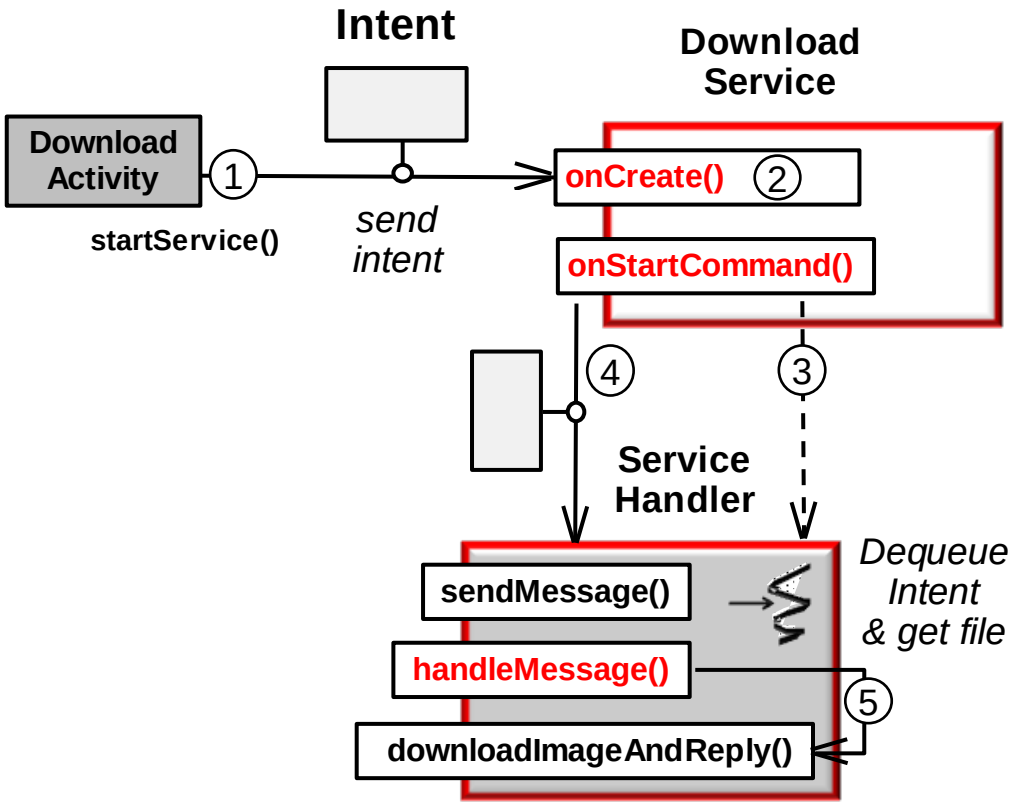
- IntentService codifies an idiom used in Android




See previous part on “Programming Started Services”

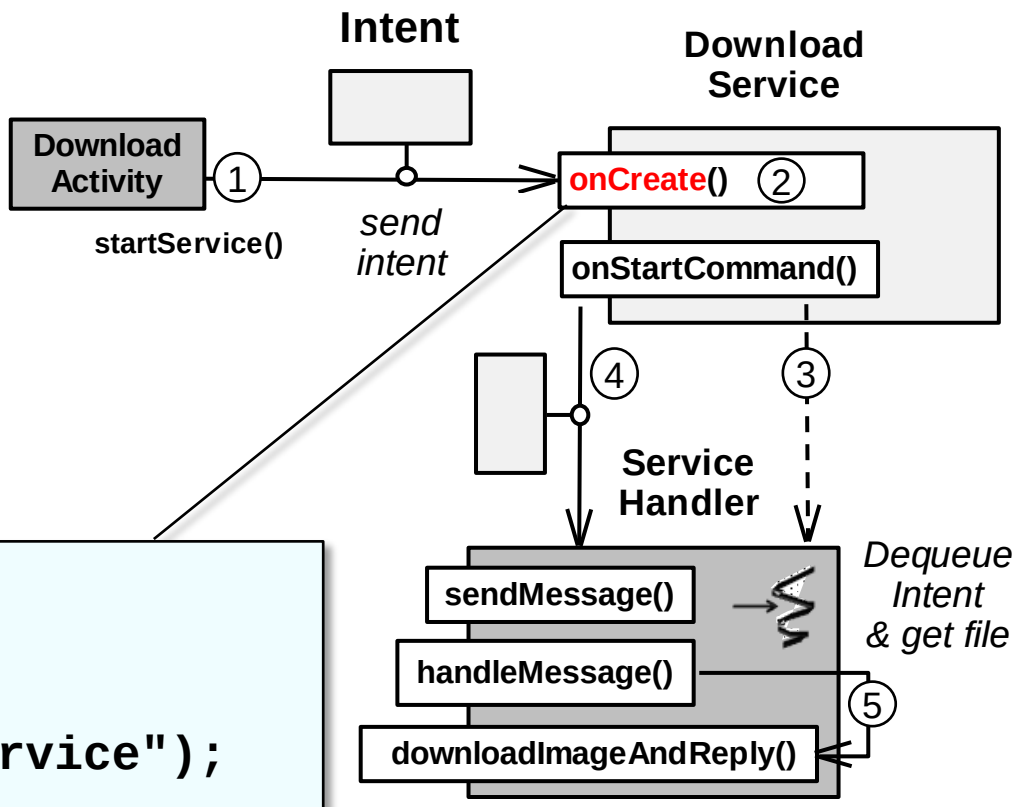
# Motivation for IntentService

- IntentService codifies an idiom used in Android



# Motivation for IntentService

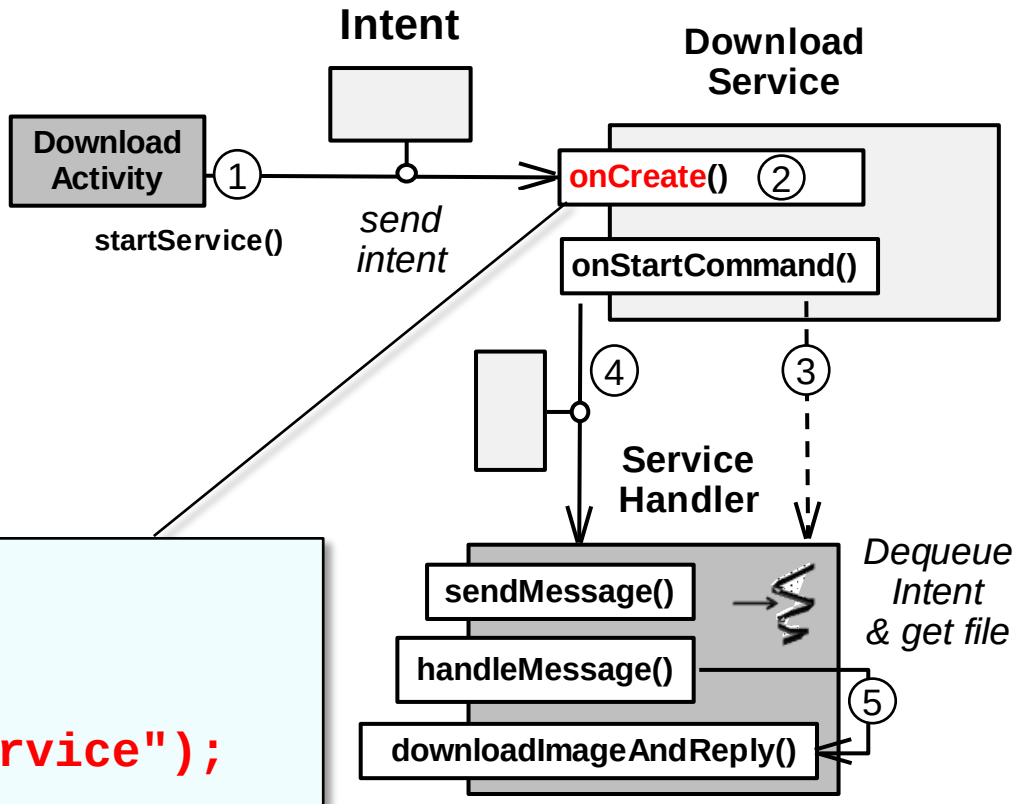
- IntentService codifies an idiom used in Android
  - Service.onCreate()




```
void onCreate() {
  ...
  HandlerThread thread = new
    HandlerThread("DownloadService");
  thread.start();

  mServiceLooper =
    thread.getLooper();
  mServiceHandler = new
    ServiceHandler(mServiceLooper);
  ...
```

# Motivation for IntentService

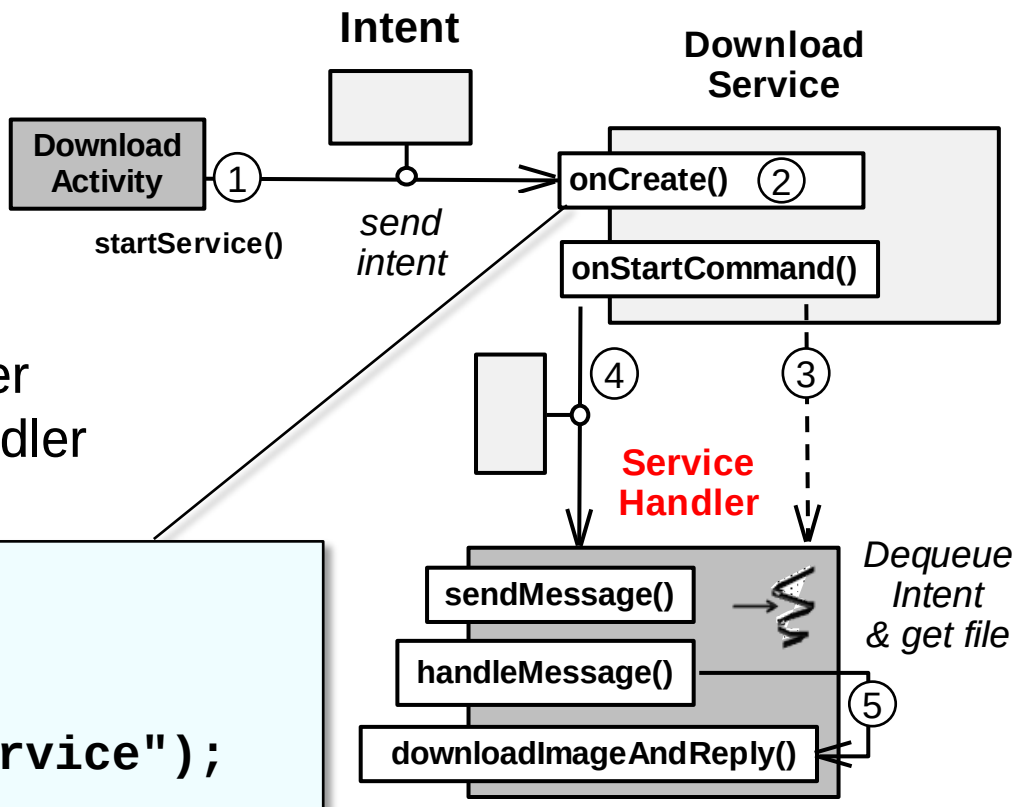
- IntentService codifies an idiom used in Android
  - Service.onCreate()
    1. Create/start a HandlerThread




```
void onCreate() {
  ...
  HandlerThread thread = new
    HandlerThread("DownloadService");
  thread.start();

  mServiceLooper =
    thread.getLooper();
  mServiceHandler = new
    ServiceHandler(mServiceLooper);
  ...
```

# Motivation for IntentService

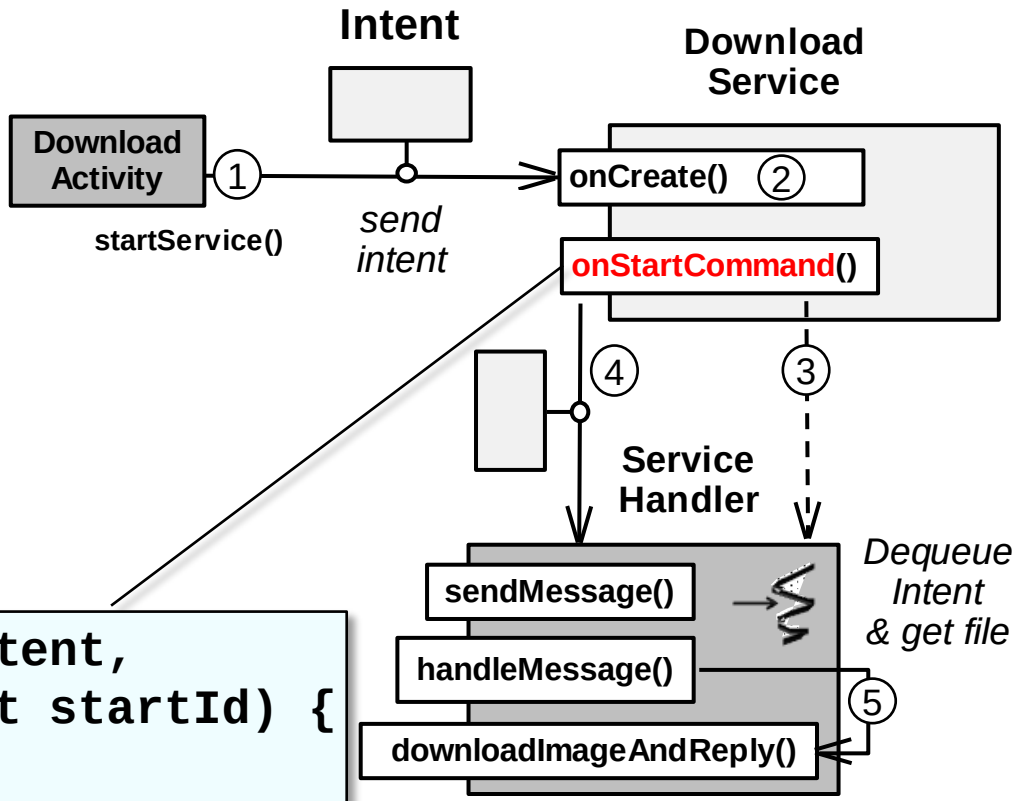
- IntentService codifies an idiom used in Android
  - Service.onCreate()
    1. Create/start a HandlerThread
    2. Give the Thread-specific Looper to an instance of a ServiceHandler




```
void onCreate() {
  ...
  HandlerThread thread = new
    HandlerThread("DownloadService");
  thread.start();

  mServiceLooper =
    thread.getLooper();
  mServiceHandler = new
    ServiceHandler(mServiceLooper);
  ...
```

# Motivation for IntentService

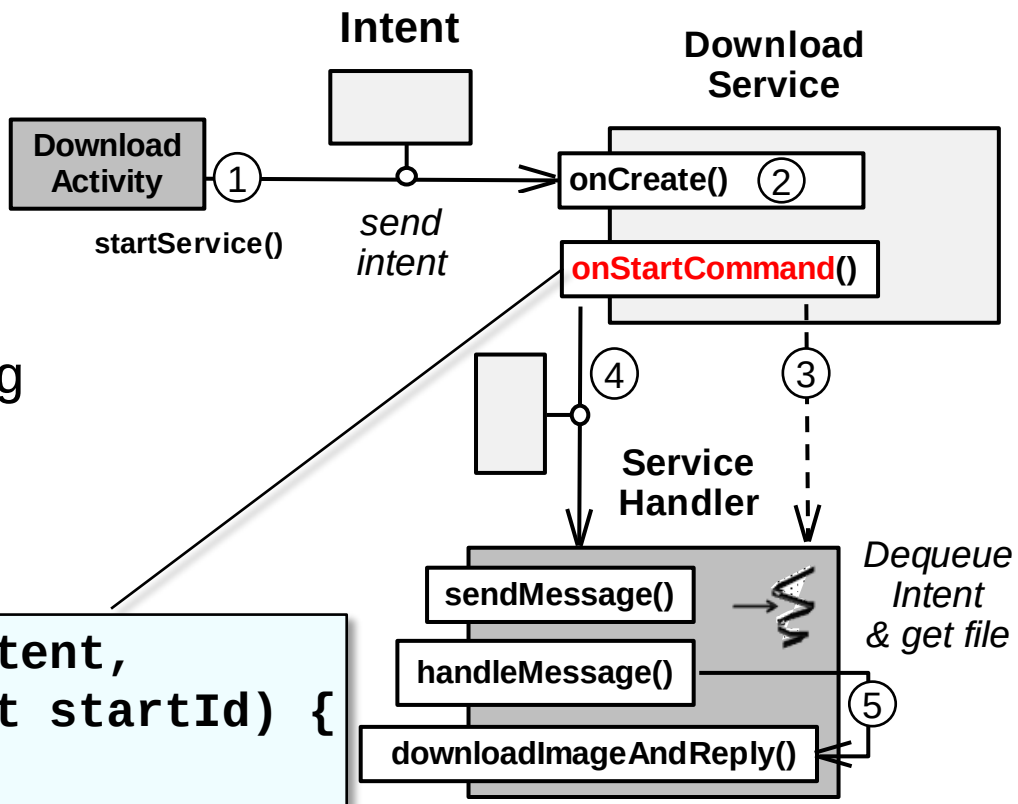
- IntentService codifies an idiom used in Android
  - Service.onCreate()
  - Service.onStartCommand()




```
int onStartCommand(Intent intent,
                   int f, int startId) {
  Message msg =
    mServiceHandler.obtainMessage();
  msg.arg1 = startId;
  msg.obj = intent;
  mServiceHandler.sendMessage(msg);
  ...
```

# Motivation for IntentService

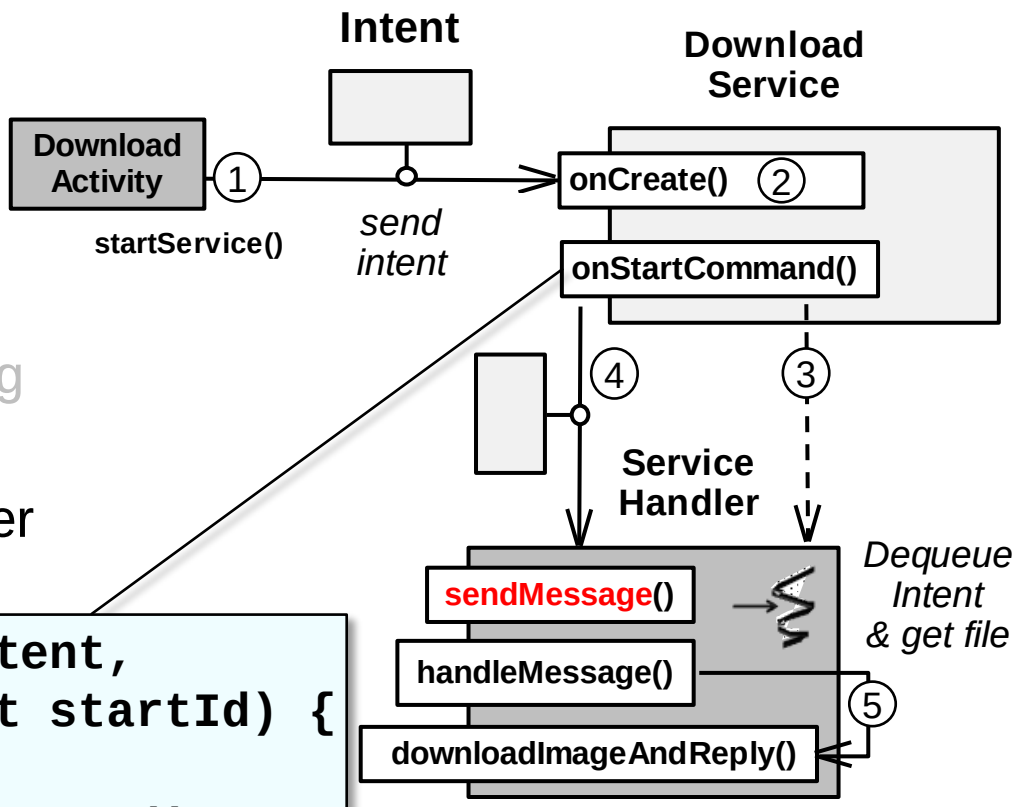
- IntentService codifies an idiom used in Android
  - Service.onCreate()
  - Service.onStartCommand()
    1. Create a Message encapsulating the Intent parameter




```
int onStartCommand(Intent intent,
                   int f, int startId) {
  Message msg =
    mServiceHandler.obtainMessage();
  msg.arg1 = startId;
  msg.obj = intent;
  mServiceHandler.sendMessage(msg);
  ...
```

# Motivation for IntentService

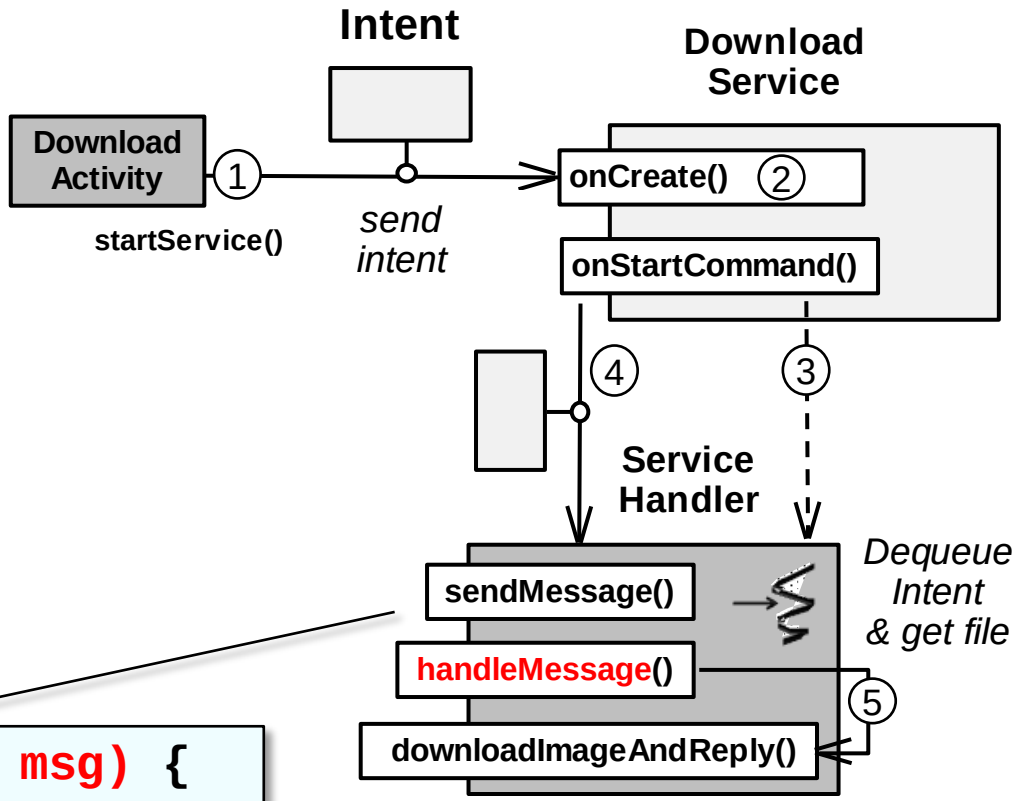
- IntentService codifies an idiom used in Android
  - Service.onCreate()
  - Service.onStartCommand()
    1. Create a Message encapsulating the Intent parameter
    2. Send Message to ServiceHandler




```
int onStartCommand(Intent intent,
                   int f, int startId) {
  Message msg =
    mServiceHandler.obtainMessage();
  msg.arg1 = startId;
  msg.obj = intent;
  mServiceHandler.sendMessage(msg);
  ...
```

# Motivation for IntentService

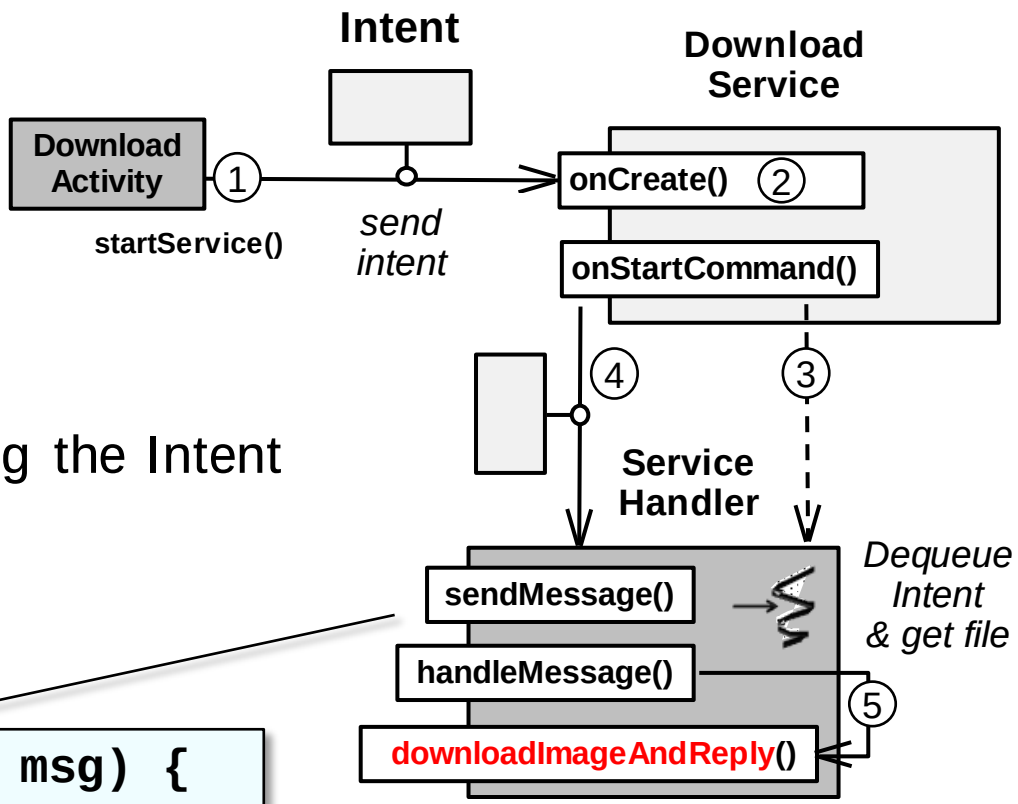
- IntentService codifies an idiom used in Android
  - Service.onCreate()
  - Service.onStartCommand()
  - ServiceHandler.handleMessage()




```
void handleMessage(Message msg) {
  downloadImageAndReply
    ((Intent) msg.obj);
  stopSelf(msg.arg1);
```

# Motivation for IntentService

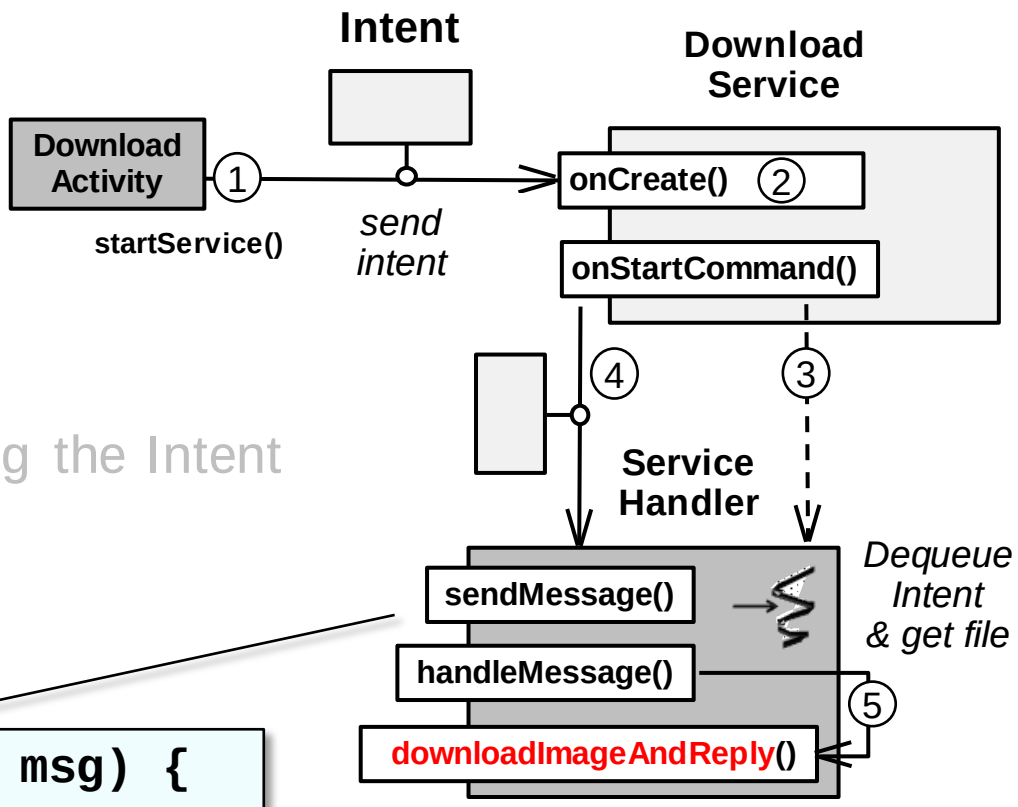
- IntentService codifies an idiom used in Android
  - Service.onCreate()
  - Service.onStartCommand()
  - ServiceHandler.handleMessage()
    1. Process the Message containing the Intent




```
void handleMessage(Message msg) {
  downloadImageAndReply
    ((Intent) msg.obj);
  stopSelf(msg.arg1);
```

# Motivation for IntentService

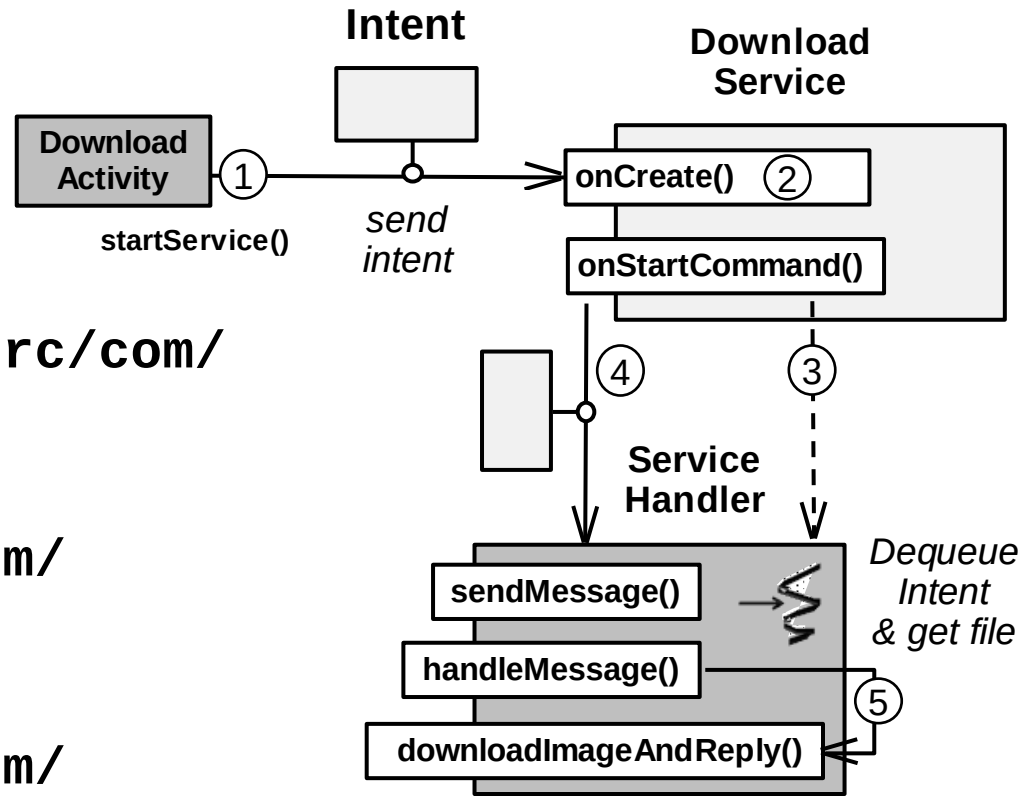
- IntentService codifies an idiom used in Android
  - Service.onCreate()
  - Service.onStartCommand()
  - ServiceHandler.handleMessage()
    1. Process the Message containing the Intent
    2. Have the Service stop itself




```
void handleMessage(Message msg) {
  downloadImageAndReply
    ((Intent) msg.obj);
  stopSelf(msg.arg1);
```

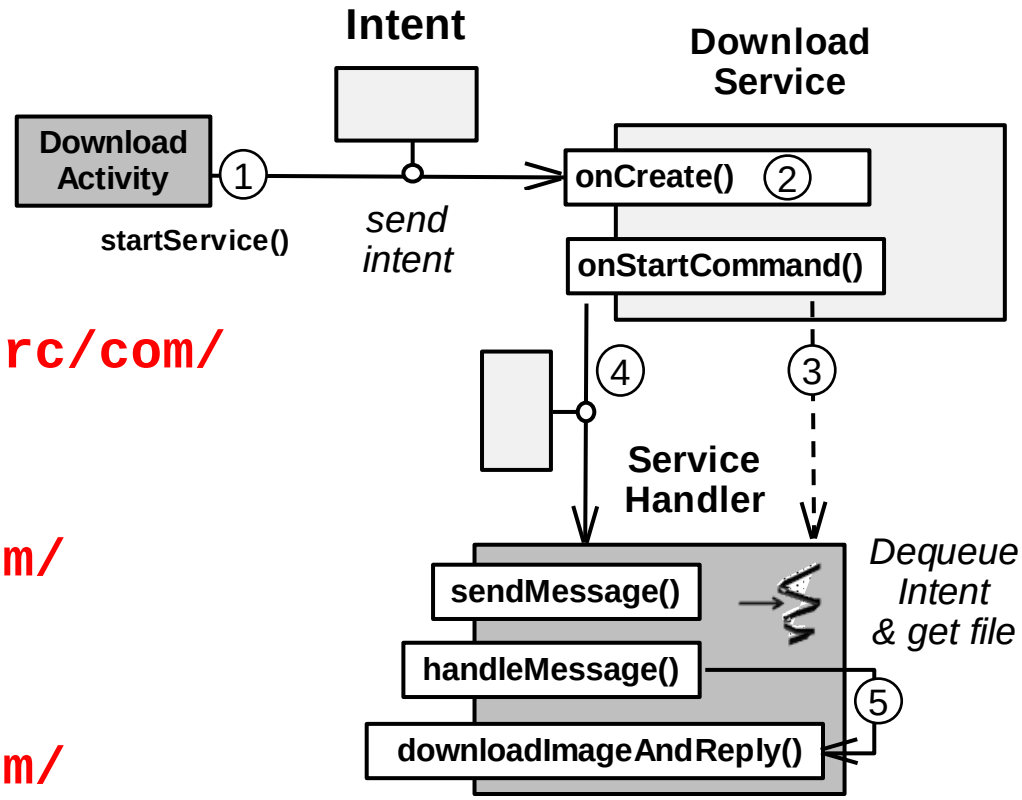
# Motivation for IntentService

- IntentService codifies an idiom used in Android
- This idioms appears in a number of packaged applications, e.g.
  - **`packages/apps/Calendar/src/com/android/calendar/alerts/AlertService.java`**
  - **`packages/apps/Mms/src/com/android/mms/transaction/SmsReceiverService.java`**
  - **`packages/apps/Mms/src/com/android/mms/transaction/TransactionService.java`**



# Motivation for IntentService

- IntentService codifies an idiom used in Android
- This idioms appears in a number of packaged applications, e.g.
  - **packages/apps/Calendar/src/com/android/calendar/alerts/AlertService.java**
  - **packages/apps/Mms/src/com/android/mms/transaction/SmsReceiverService.java**
  - **packages/apps/Mms/src/com/android/mms/transaction/TransactionService.java**



# Overview of the IntentService (Part 1)

# Overview of IntentService

- IntentService codifies the idiom for concurrent processing of Intents into a framework

```
public class IntentService
             extends Service {
  ...

  protected abstract void
    onHandleIntent(Intent intent);
}
```

# Overview of IntentService

- IntentService codifies the idiom for concurrent processing of Intents into a framework

```
public class IntentService
             extends Service {
  ...

  protected abstract void
    onHandleIntent(Intent intent);
}
```

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

# Overview of IntentService

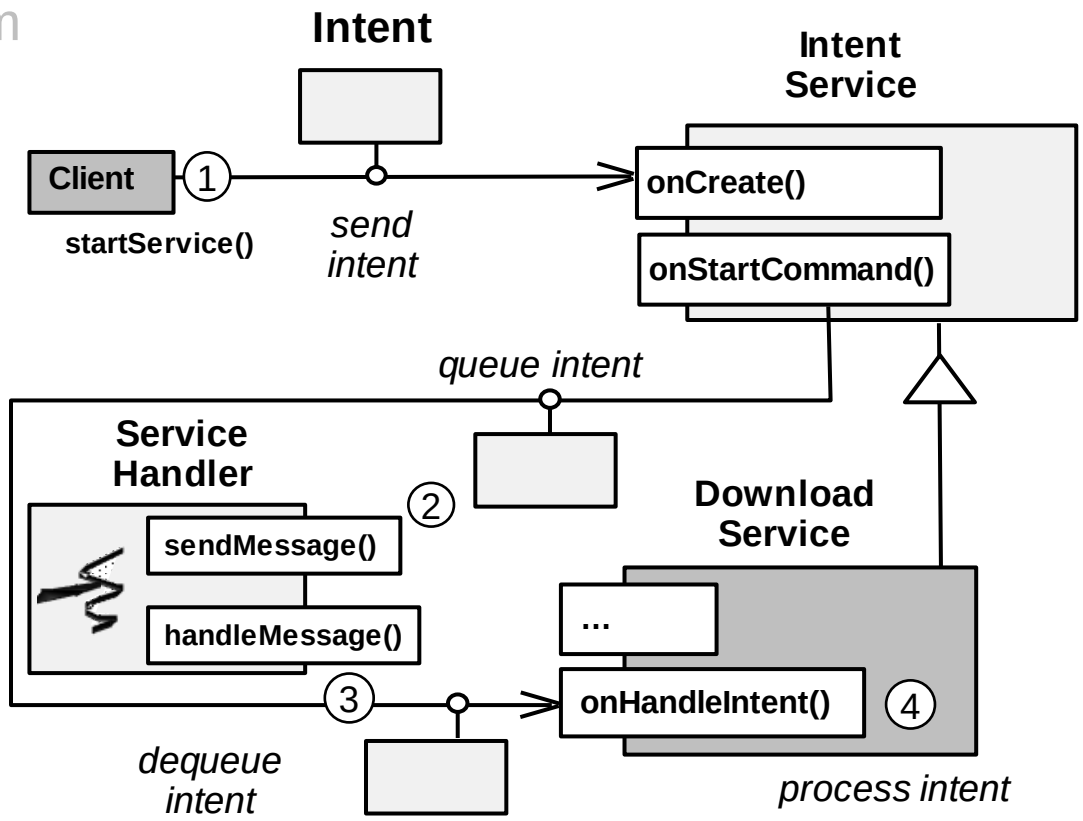
- IntentService codifies the idiom for concurrent processing of Intents into a framework
  - It's *very* easy to use

```
public class IntentService
             extends Service {
  ...

  protected abstract void
    onHandleIntent(Intent intent);
}
```

*Subclasses simply override this hook method to process an Intent in a single background Thread*

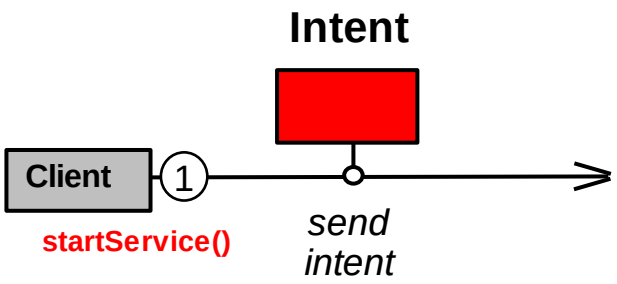
# Overview of IntentService

- IntentService codifies the idiom for concurrent processing of Intents into a framework
- Clients send Intents via calls to startService()



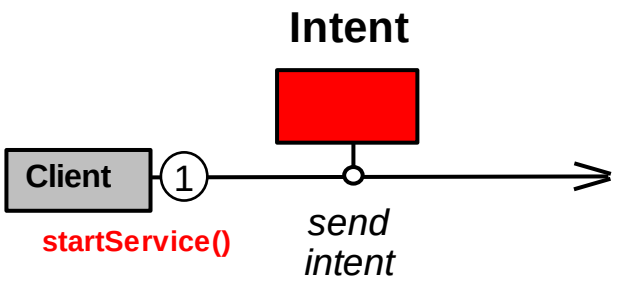
# Overview of IntentService

- IntentService codifies the idiom for concurrent processing of Intents into a framework
- Clients send Intents via calls to startService()



# Overview of IntentService

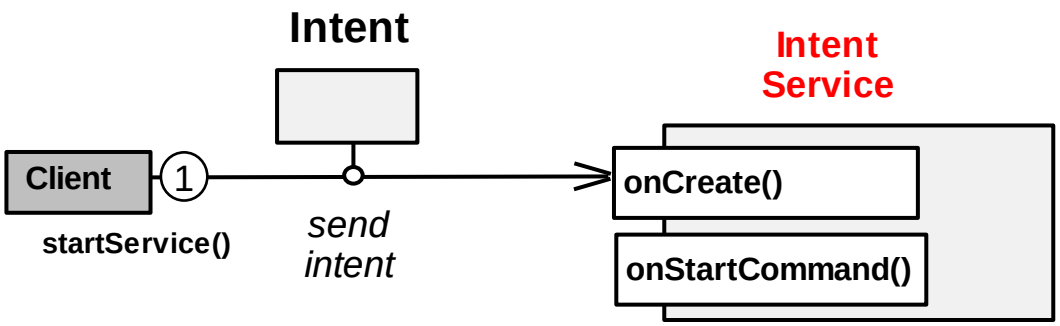
- IntentService codifies the idiom for concurrent processing of Intents into a framework
- Clients send Intents via calls to startService()
  - Data & objects can be passed to the Service by putting “extras” into Intents




www.itcuties.com/android/intent-putextra has more info on extras

# Overview of IntentService

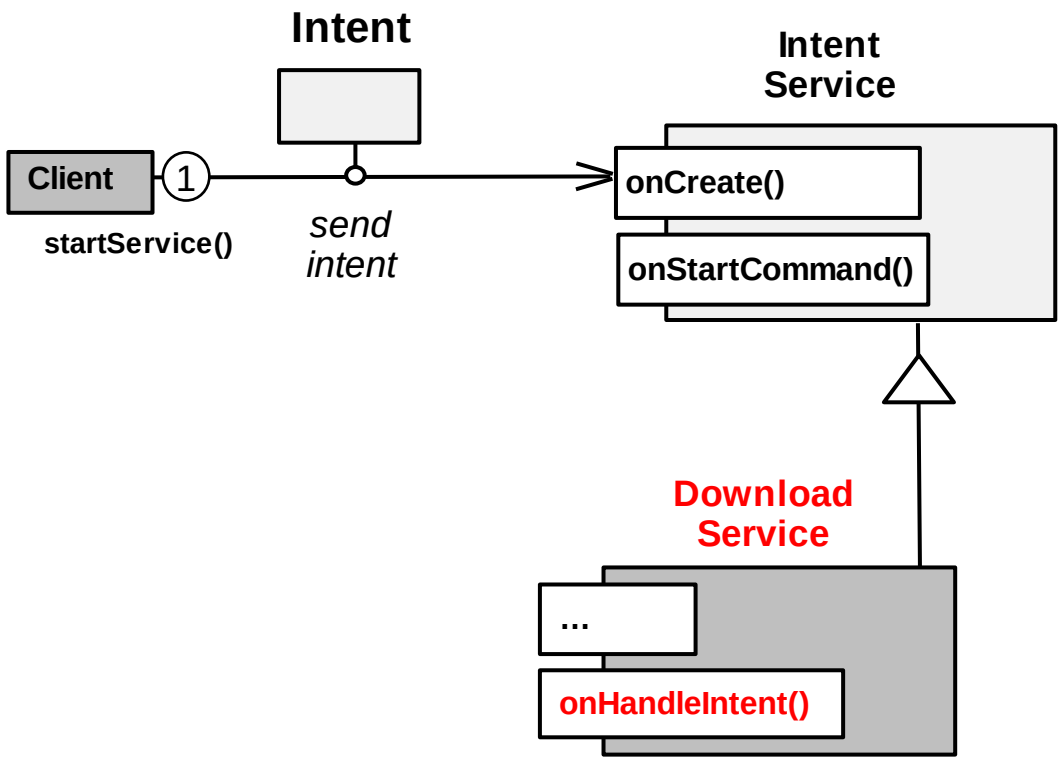
- IntentService codifies the idiom for concurrent processing of Intents into a framework
- Clients send Intents via calls to startService()
  - Data & objects can be passed to the Service by putting "extras" into Intents
  - The IntentService is launched on-demand via the *Activator* pattern




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

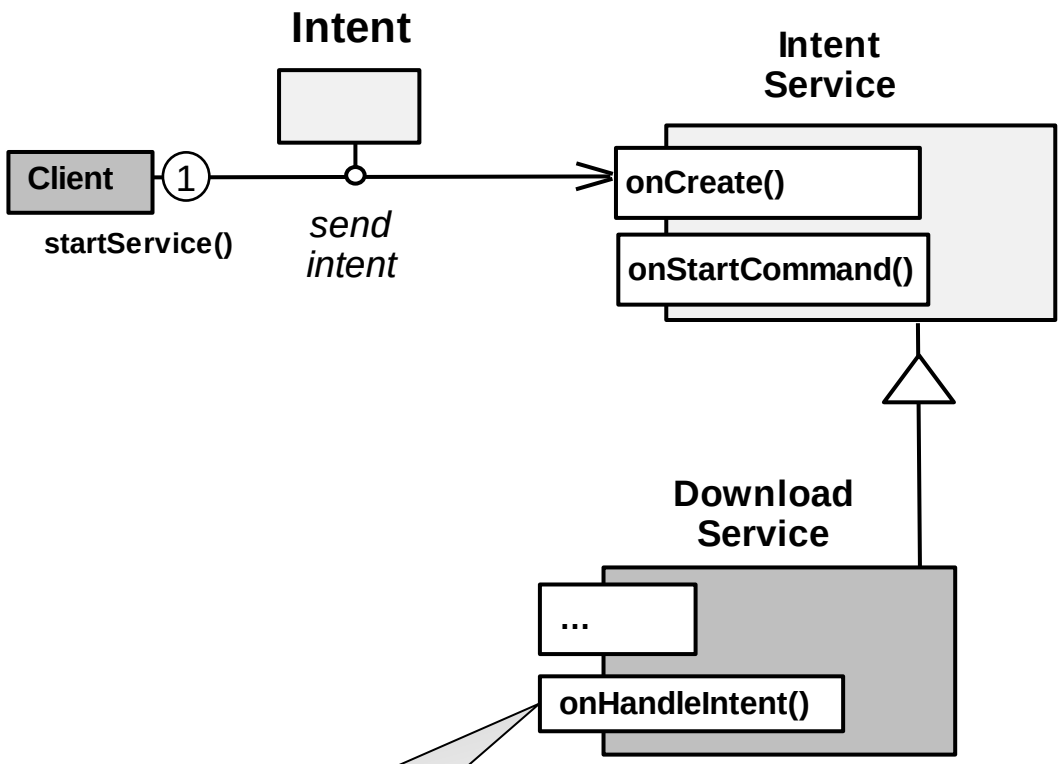
# Overview of IntentService

- IntentService codifies the idiom for concurrent processing of Intents into a framework
- Clients send Intents via calls to startService()
- A subclass of IntentService implements onHandleIntent()
  - This hook method processes the Intent sent by the client in a background Thread



# Overview of IntentService

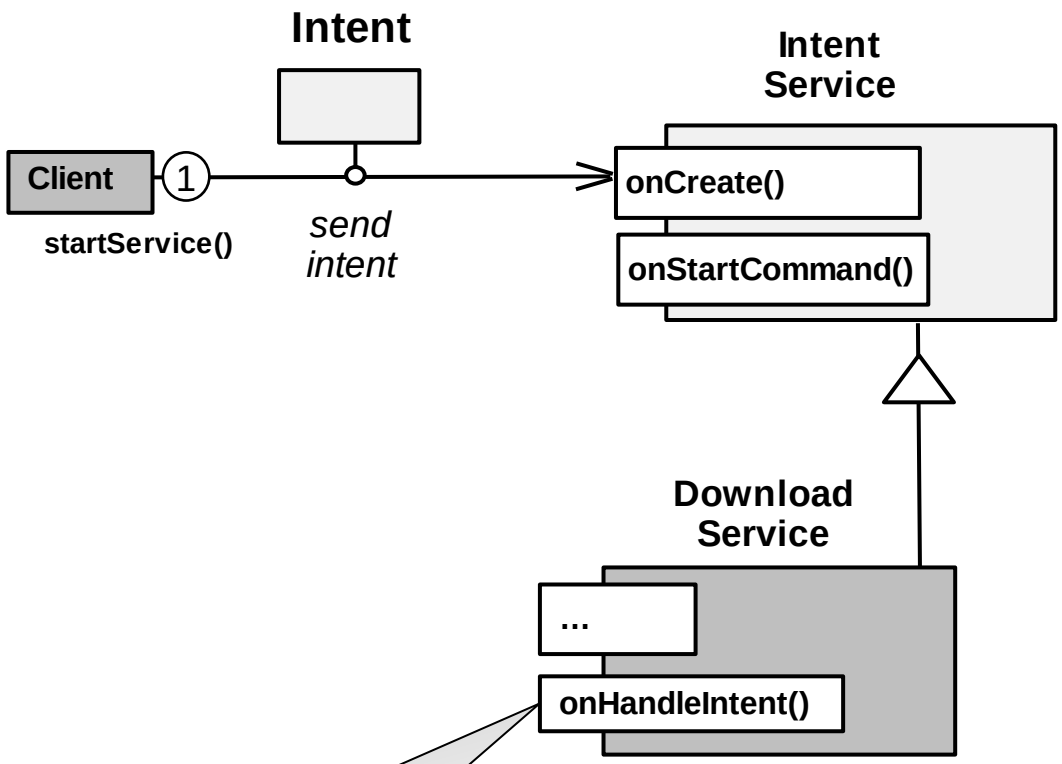
- IntentService codifies the idiom for concurrent processing of Intents into a framework
- Clients send Intents via calls to startService()
- A subclass of IntentService implements onHandleIntent()
  - This hook method processes the Intent sent by the client in a background Thread




```
void onHandleIntent(Intent intent) {
  downloadImageAndReply(intent);
}
```

# Overview of IntentService

- IntentService codifies the idiom for concurrent processing of Intents into a framework
- Clients send Intents via calls to startService()
- A subclass of IntentService implements onHandleIntent()
  - This hook method processes the Intent sent by the client in a background Thread

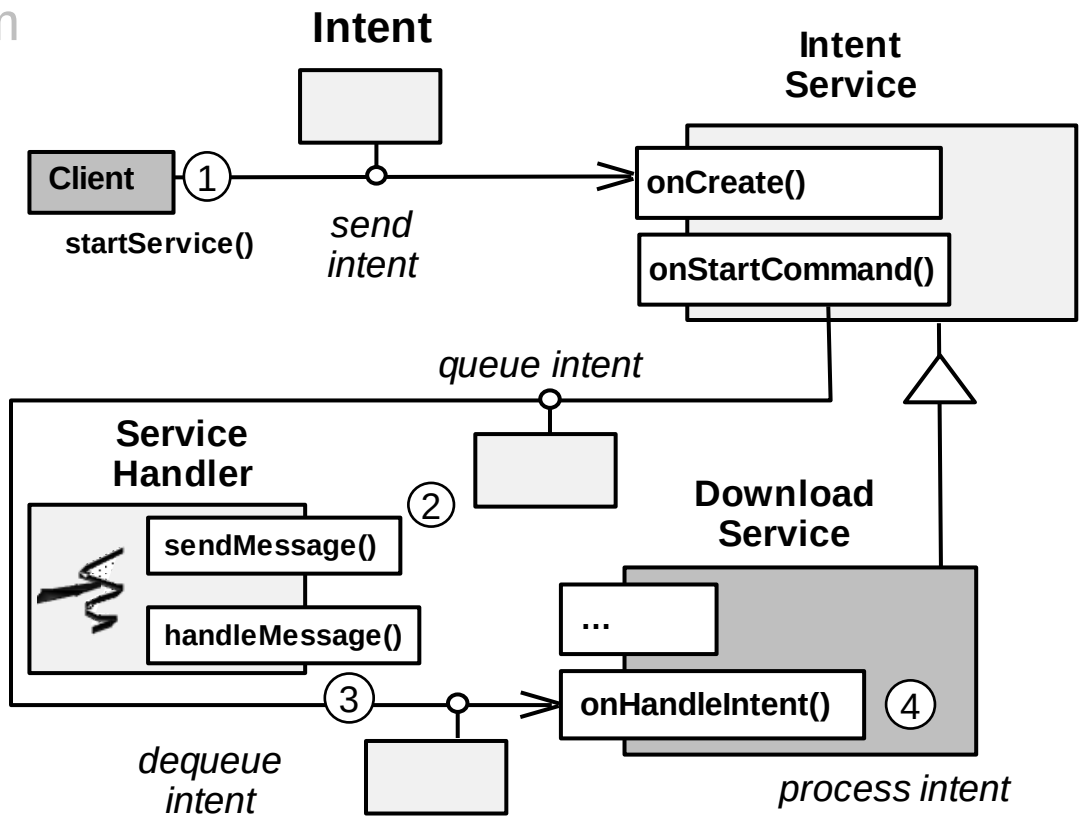



```
void onHandleIntent(Intent intent) {
  downloadImageAndReply(intent);
}
```

# Overview of the IntentService (Part 2)

# Overview of IntentService

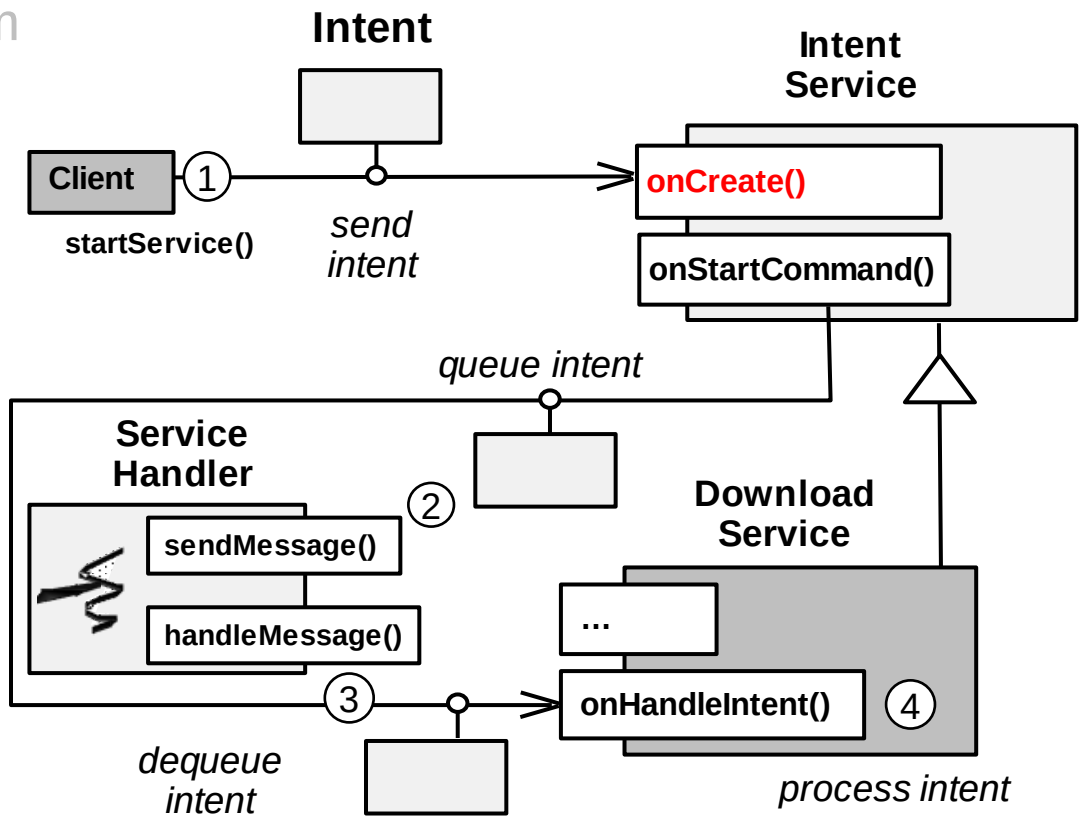
- IntentService codifies the idiom for concurrent processing of Intents into a framework
- Clients send Intents via calls to startService()
- A subclass of IntentService implements onHandleIntent()
- The IntentService does several things




frameworks/base/core/java/android/app/IntentService.java has the code

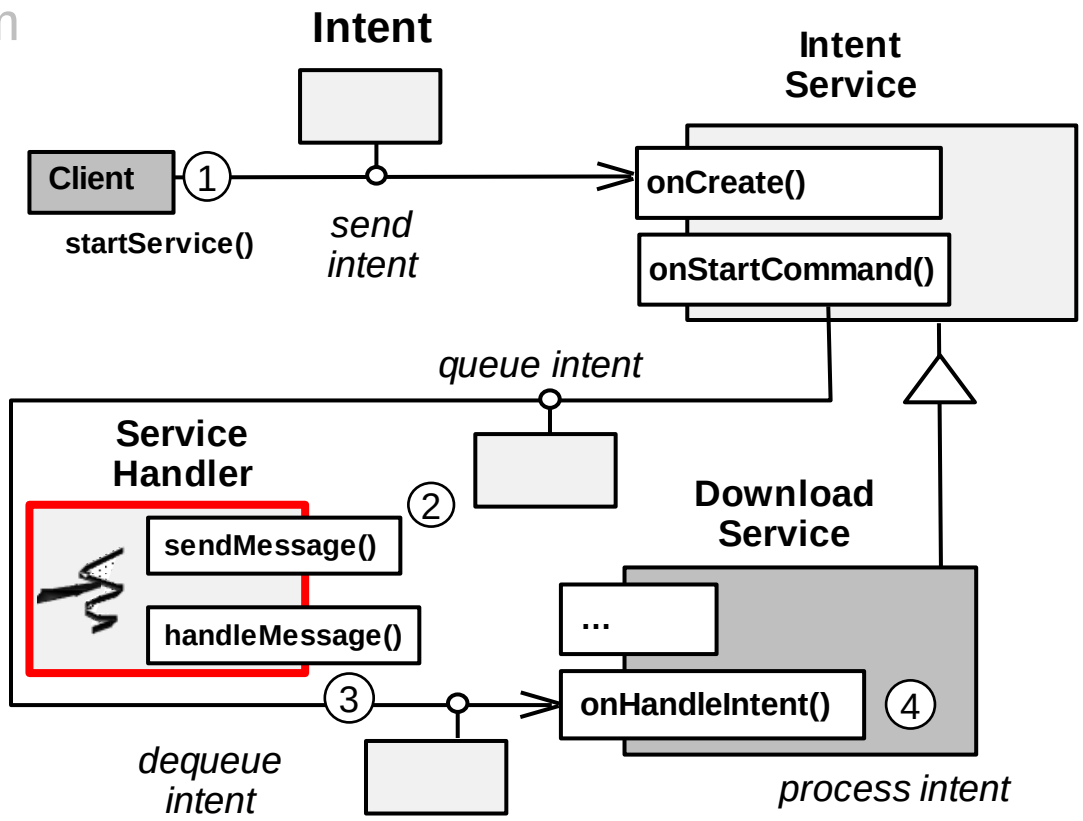
# Overview of IntentService

- IntentService codifies the idiom for concurrent processing of Intents into a framework
- Clients send Intents via calls to startService()
- A subclass of IntentService implements onHandleIntent()
- The IntentService does several things
  - Creates a ServiceHandler



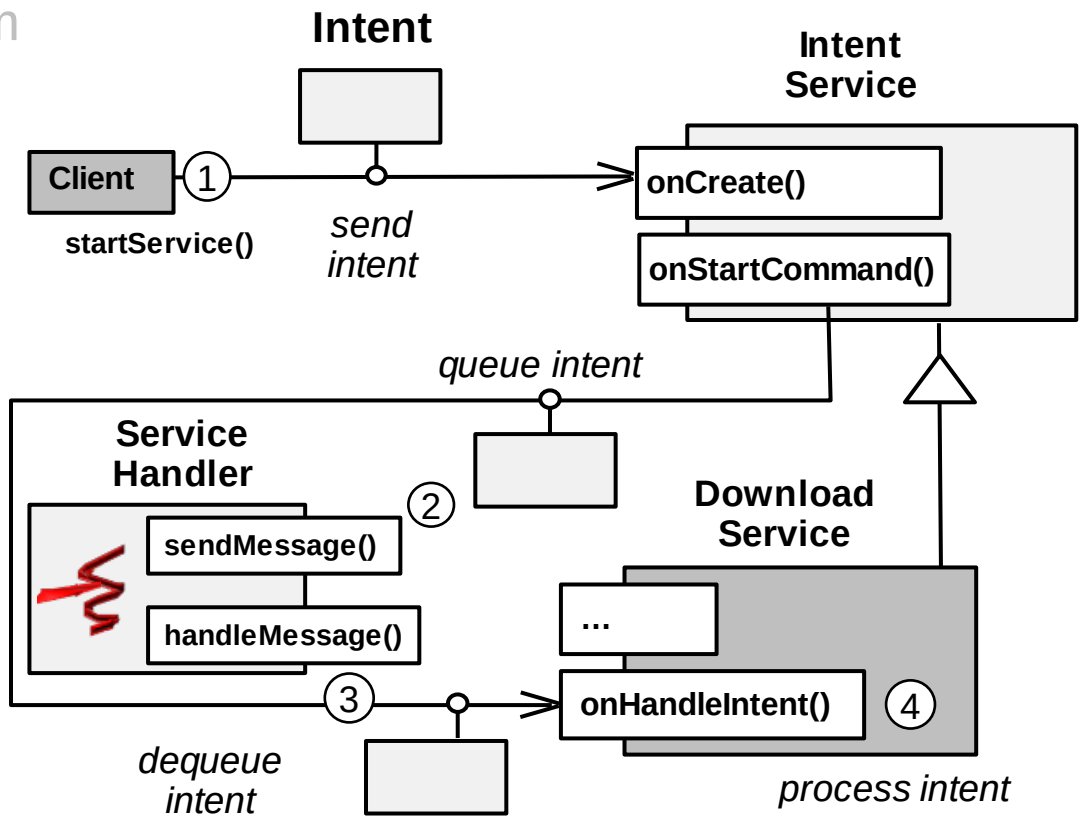
# Overview of IntentService

- IntentService codifies the idiom for concurrent processing of Intents into a framework
- Clients send Intents via calls to startService()
- A subclass of IntentService implements onHandleIntent()
- The IntentService does several things
  - Creates a ServiceHandler



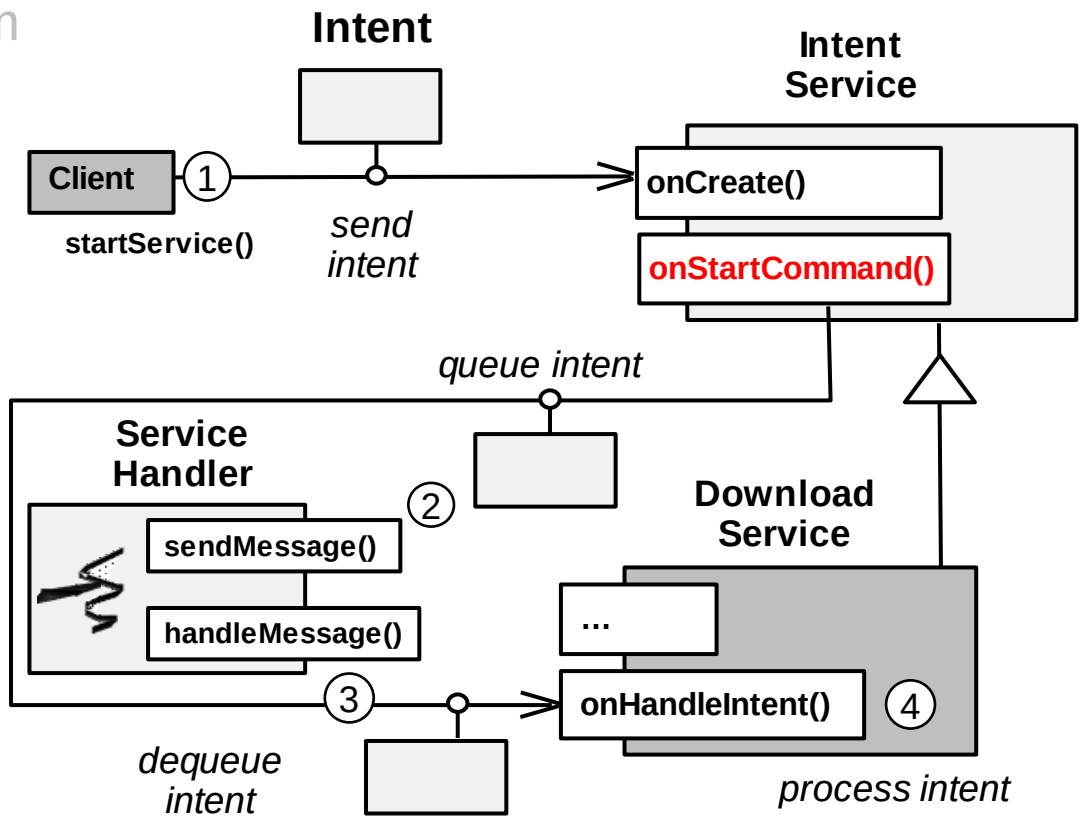
# Overview of IntentService

- IntentService codifies the idiom for concurrent processing of Intents into a framework
- Clients send Intents via calls to startService()
- A subclass of IntentService implements onHandleIntent()
- The IntentService does several things
  - Creates a ServiceHandler
    - Internally creates a single background thread



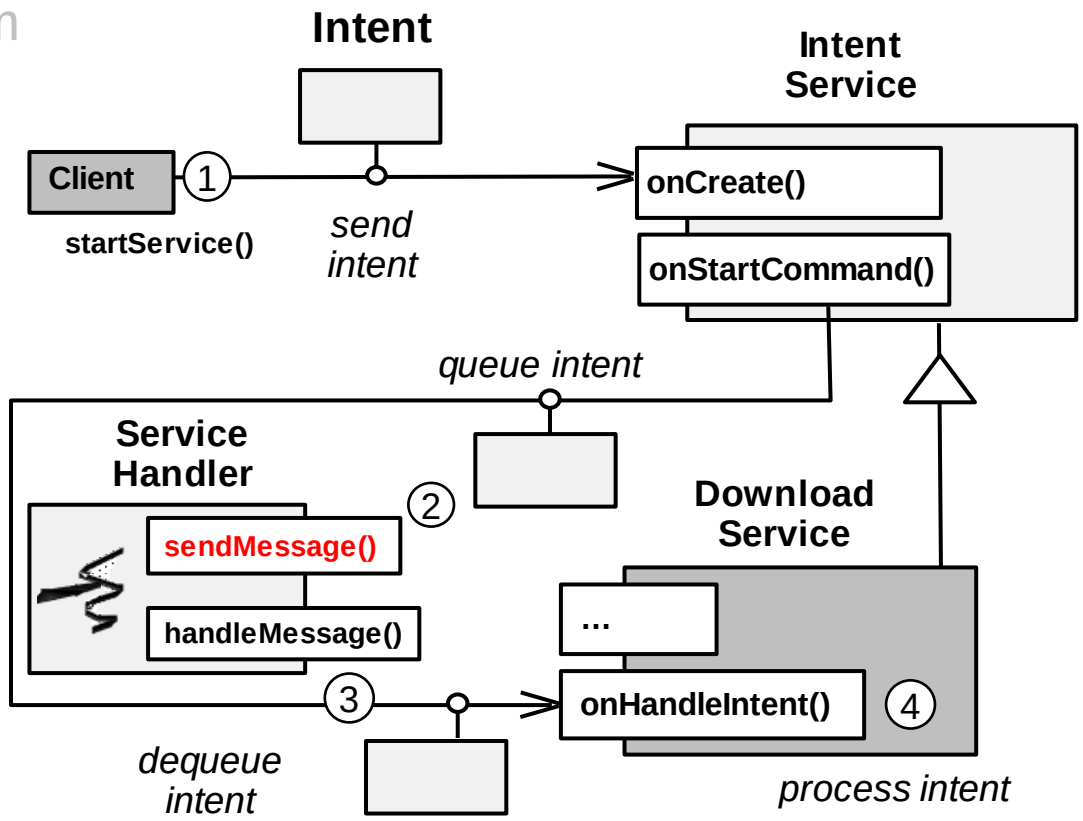
# Overview of IntentService

- IntentService codifies the idiom for concurrent processing of Intents into a framework
- Clients send Intents via calls to startService()
- A subclass of IntentService implements onHandleIntent()
- The IntentService does several things
  - Creates a ServiceHandler
  - Receives & queues Intents in ServiceHandler



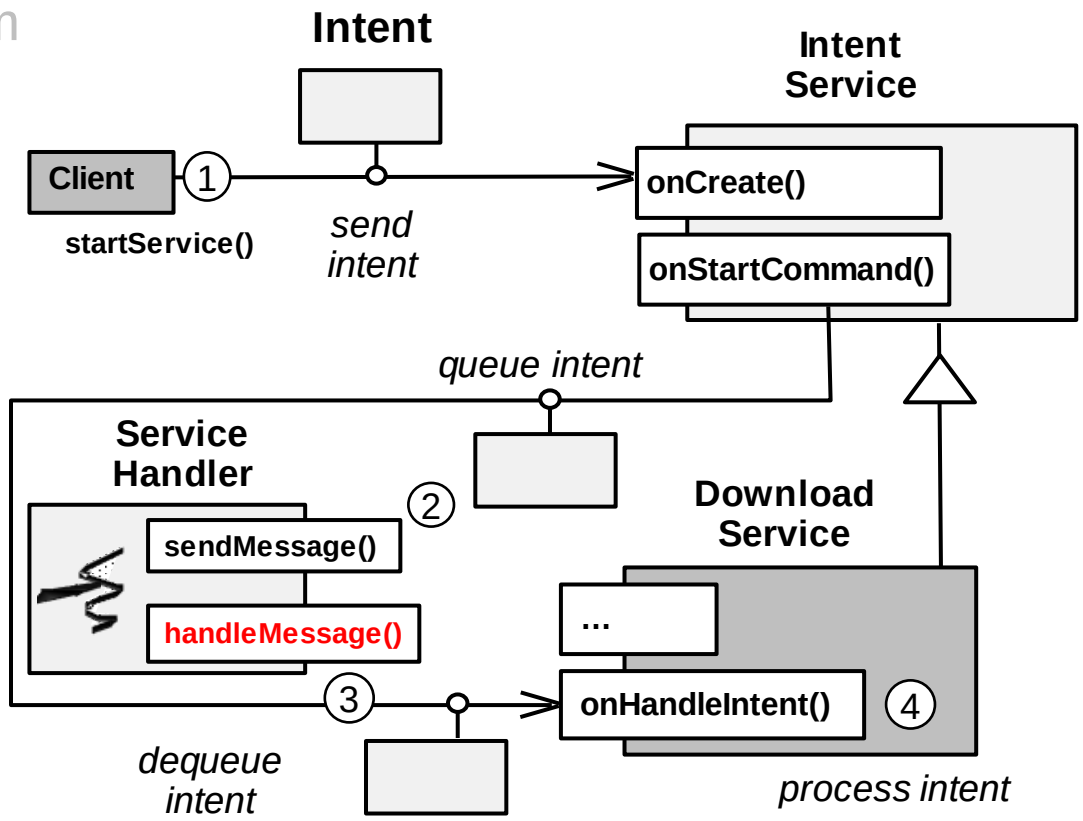
# Overview of IntentService

- IntentService codifies the idiom for concurrent processing of Intents into a framework
- Clients send Intents via calls to startService()
- A subclass of IntentService implements onHandleIntent()
- The IntentService does several things
  - Creates a ServiceHandler
  - Receives & queues Intents in ServiceHandler



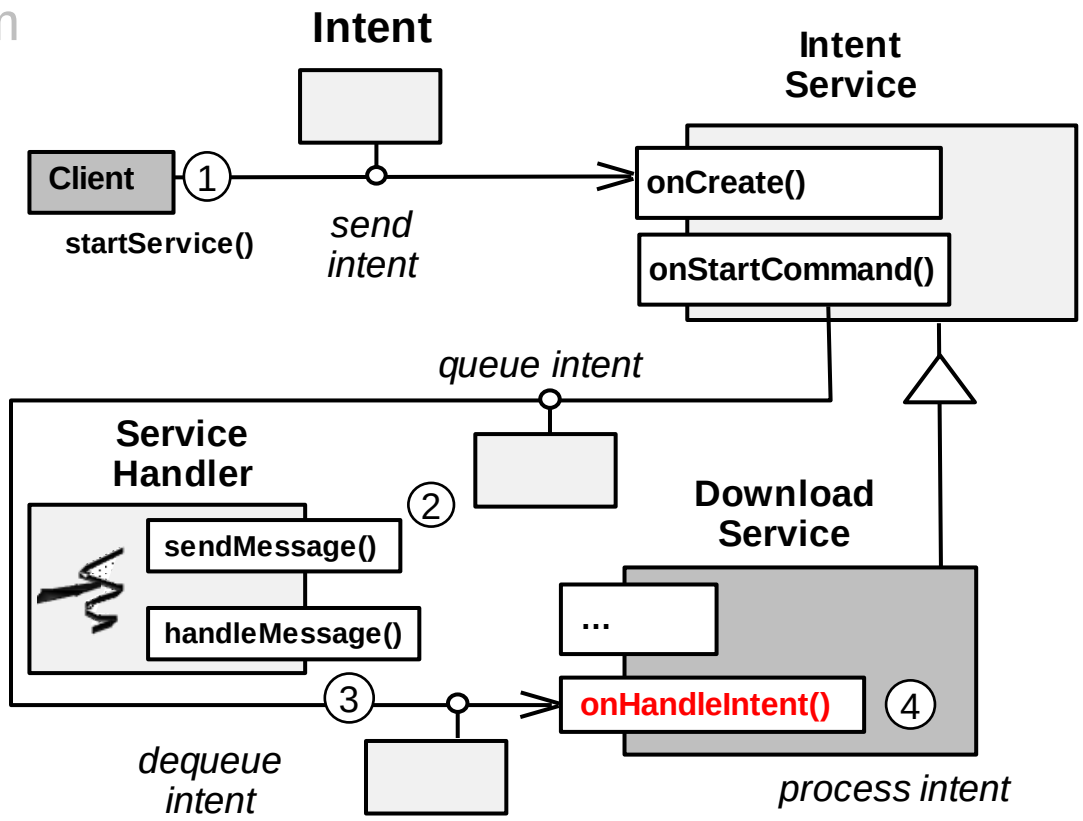
# Overview of IntentService

- IntentService codifies the idiom for concurrent processing of Intents into a framework
- Clients send Intents via calls to startService()
- A subclass of IntentService implements onHandleIntent()
- The IntentService does several things
  - Creates a ServiceHandler
  - Receives & queues Intents in ServiceHandler
  - Processes the Intents "in the background"



# Overview of IntentService

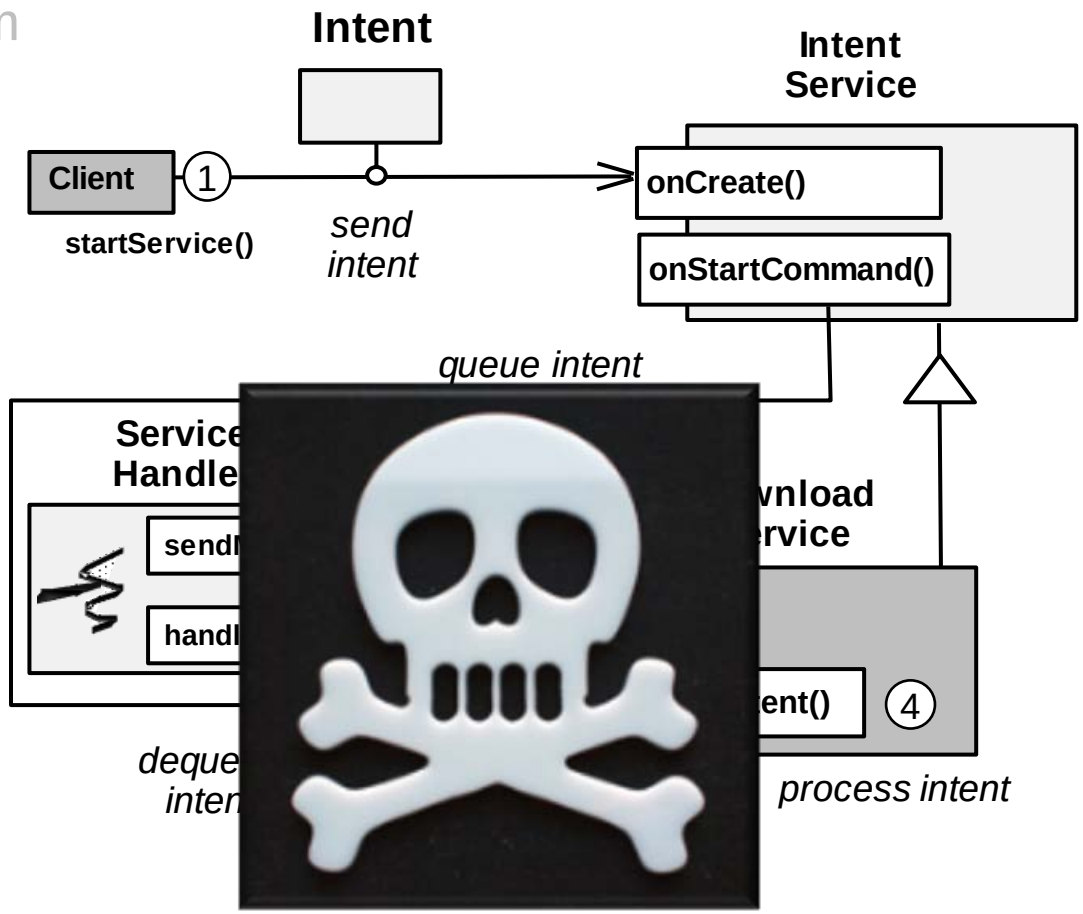
- IntentService codifies the idiom for concurrent processing of Intents into a framework
- Clients send Intents via calls to startService()
- A subclass of IntentService implements onHandleIntent()
- The IntentService does several things
  - Creates a ServiceHandler
  - Receives & queues Intents in ServiceHandler
  - Processes the Intents "in the background"




Only one Intent processed at a time by the one & only Thread

# Overview of IntentService

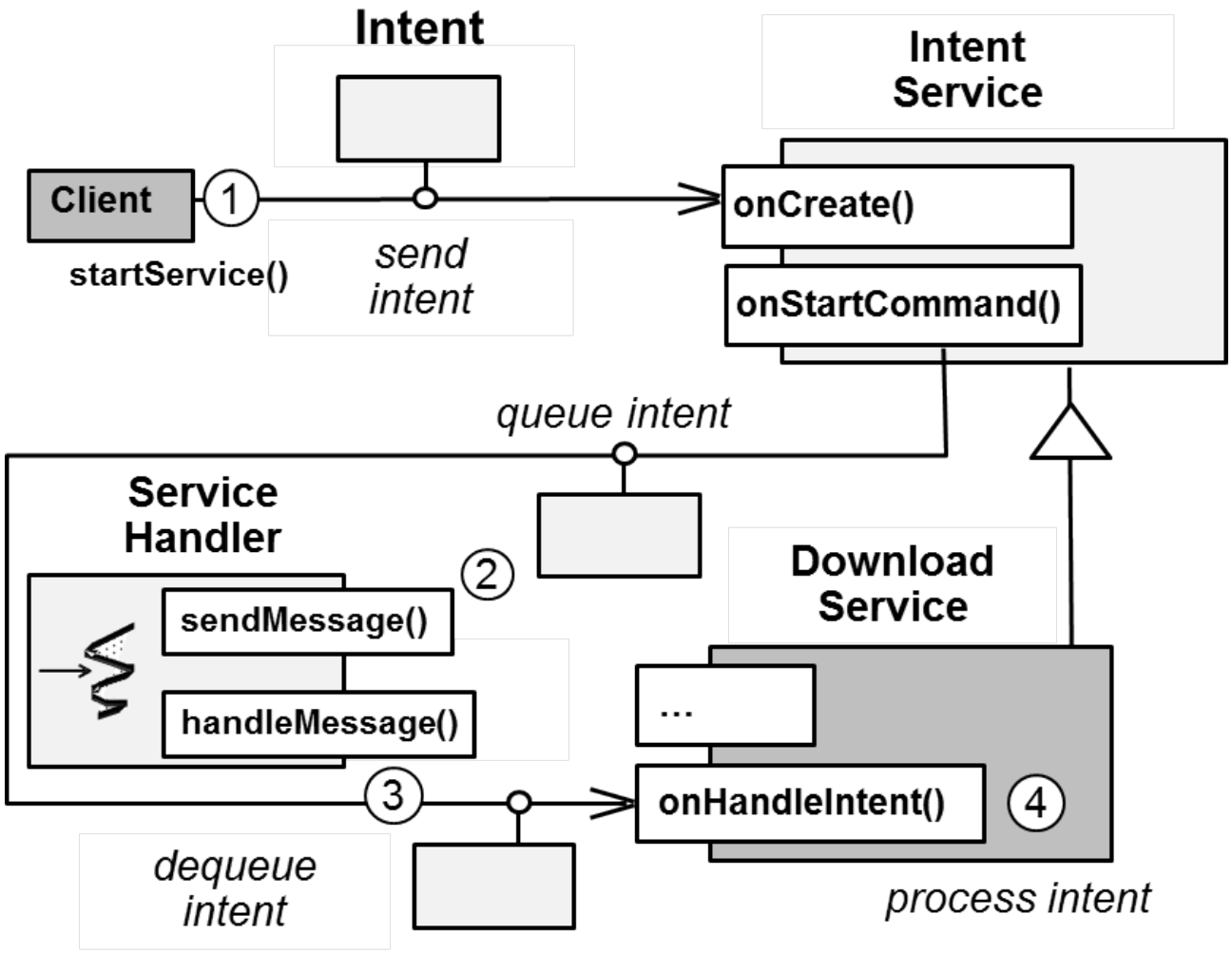
- IntentService codifies the idiom for concurrent processing of Intents into a framework
- Clients send Intents via calls to startService()
- A subclass of IntentService implements onHandleIntent()
- The IntentService does several things
  - Creates a ServiceHandler
  - Receives & queues Intents in ServiceHandler
  - Processes the Intents "in the background"
  - Stops Service when there are no more Intents to process

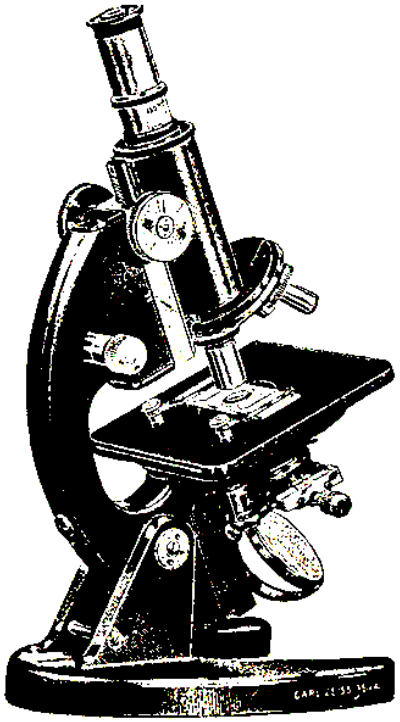



In contrast, a Service must stop itself manually via stopSelf()

# The IntentService Implementation

# Implementation of IntentService



frameworks/base/core/java/android/app/IntentService.java has the code

# Implementation of IntentService

```
public class IntentService extends Service {
   ...
```

Base class for Services that handle asynchronous requests (expressed as Intents) on demand

# Implementation of IntentService

```
public class IntentService extends Service {
   ...
```

Base class for Services that handle asynchronous requests (expressed as Intents) on demand

# Implementation of IntentService

```
public class IntentService extends Service {
  ...
  private volatile Looper mServiceLooper;
  private volatile ServiceHandler mServiceHandler;

  public void onCreate() {
    super.onCreate();



    HandlerThread thread = new HandlerThread("IntentService["
                                             + mName + "]");
    thread.start();


    mServiceLooper = thread.getLooper();
    mServiceHandler = new ServiceHandler(mServiceLooper);
 }
```

**Useful data members**

The volatile keyword ensures data members are properly visible to Threads

# Implementation of IntentService

```
public class IntentService extends Service {
  ...
  private volatile Looper mServiceLooper;
  private volatile ServiceHandler mServiceHandler;

  public void onCreate() {
    super.onCreate();

    HandlerThread thread = new HandlerThread("IntentService["
                                             + mName + "]");
    thread.start();


    mServiceLooper = thread.getLooper();
    mServiceHandler = new ServiceHandler(mServiceLooper);
 }
```

**Called when IntentService is first created**

# Implementation of IntentService

```
public class IntentService extends Service {
  ...
  private volatile Looper mServiceLooper;
  private volatile ServiceHandler mServiceHandler;

  public void onCreate() {
    super.onCreate();

    HandlerThread thread = new HandlerThread("IntentService["
                                             + mName + "]");
    thread.start();


    mServiceLooper = thread.getLooper();
    mServiceHandler = new ServiceHandler(mServiceLooper);
 }
```

**Create/start a separate HandlerThread to process the Intent concurrently in the background**

# Implementation of IntentService

```
public class IntentService extends Service {
  ...
  private volatile Looper mServiceLooper;
  private volatile ServiceHandler mServiceHandler;

  public void onCreate() {
    super.onCreate();

    HandlerThread thread = new HandlerThread("IntentService["
                                             + mName + "]");
    thread.start();


    mServiceLooper = thread.getLooper();
    mServiceHandler = new ServiceHandler(mServiceLooper);
 }
```

**Create/start a separate HandlerThread to process the Intent concurrently in the background**

See earlier part on “Android Looper”

# Implementation of IntentService

```
public class IntentService extends Service {
  ...
  private volatile Looper mServiceLooper;
  private volatile ServiceHandler mServiceHandler;

  public void onCreate() {
    super.onCreate();



    HandlerThread thread = new HandlerThread ("IntentService["
                                              + mName + "]");
    thread.start();
```

Get the HandlerThread's Looper & use it for our Handler

```
    mServiceLooper = thread.getLooper();
    mServiceHandler = new ServiceHandler(mServiceLooper);
 }
```

# Implementation of IntentService

```
public class IntentService extends Service {
  ...
  private volatile Looper mServiceLooper;
  private volatile ServiceHandler mServiceHandler;

  public void onCreate() {
    super.onCreate();



    HandlerThread thread = new HandlerThread ("IntentService["
                                              + mName + "]");
    thread.start();
```

Get the HandlerThread's Looper & use it for our Handler

```
    mServiceLooper = thread.getLooper();
    mServiceHandler = new ServiceHandler(mServiceLooper);
 }
```

# Implementation of IntentService

```
public class IntentService extends Service {
  ...
  public int onStartCommand(Intent intent, int f, int startId) {
    onStart(intent, startId);
    return mRedelivery ? START_REDELIVER_INTENT
                       : START_NOT_STICKY;
  }

  public void onStart(Intent intent, int startId) {
    Message msg = mServiceHandler.obtainMessage();
    msg.arg1 = startId;
    msg.obj = intent;


    mServiceHandler.sendMessage(msg);
  }
  ...
```

Called each time a Started Service is sent an Intent via startService()

# Implementation of IntentService

```
public class IntentService extends Service {
  ...
  public int onStartCommand(Intent intent, int f, int startId) {
    onStart(intent, startId);
    return mRedelivery ? START_REDELIVER_INTENT
                       : START_NOT_STICKY;
  }

  public void onStart(Intent intent, int startId) {
    Message msg = mServiceHandler.obtainMessage();
    msg.arg1 = startId;
    msg.obj = intent;


    mServiceHandler.sendMessage(msg);
  }
  ...
```

Helper method

# Implementation of IntentService

```
public class IntentService extends Service {
  ...
  public int onStartCommand(Intent intent, int f, int startId) {


    onStart(intent, startId);
    return mRedelivery ? START_REDELIVER_INTENT
                       : START_NOT_STICKY;
  }

  public void onStart(Intent intent, int startId)
    Message msg = mServiceHandler.obtainMessage();
    msg.arg1 = startId;
    msg.obj = intent;


    mServiceHandler.sendMessage(msg);
  }
  ...
```

**Forwards Intent to the ServiceHandler**

# Implementation of IntentService

```
public class IntentService extends Service {
  ...
  public int onStartCommand(Intent intent, int f, int startId) {


    onStart(intent, startId);
    return mRedelivery ? START_REDELIVER_INTENT
                       : START_NOT_STICKY;
  }

  public void onStart(Intent intent, int startId)
    Message msg = mServiceHandler.obtainMessage();
    msg.arg1 = startId;
    msg.obj = intent;


    mServiceHandler.sendMessage(msg);
  }
  ...
```

Create a Message

# Implementation of IntentService

```
public class IntentService extends Service {
  ...
  public int onStartCommand(Intent intent, int f, int startId) {


    onStart(intent, startId);
    return mRedelivery ? START_REDELIVER_INTENT
                       : START_NOT_STICKY;
  }

  public void onStart(Intent intent, int startId) {
    Message msg = mServiceHandler.obtainMessage();
    msg.arg1 = startId;
    msg.obj = intent;
```

**Include Intent & start ID in Message to guide subsequent processing & shutdown**

```
    mServiceHandler.sendMessage(msg);
  }
  ...
```

# Implementation of IntentService

```
public class IntentService extends Service {
  ...
  public int onStartCommand(Intent intent, int f, int startId) {


    onStart(intent, startId);
    return mRedelivery ? START_REDELIVER_INTENT
                       : START_NOT_STICKY;
  }

  public void onStart(Intent intent, int startId) {
    Message msg = mServiceHandler.obtainMessage();
    msg.arg1 = startId;
    msg.obj = intent;

    mServiceHandler.sendMessage(msg);
  }
  ...
```

**Send Message to ServiceHandler for processing in the background Thread**

# Implementation of IntentService

```
public class IntentService extends Service {
  ...

  private final class ServiceHandler extends Handler {
    public ServiceHandler(Looper looper) { super(looper); }

    public void handleMessage(Message msg) {
      onHandleIntent((Intent) msg.obj);

      stopSelf(msg.arg1);
    }

  }
  ...
```

Receives Messages passed via sendMessage()

# Implementation of IntentService

```
public class IntentService extends Service {
  ...

  private final class ServiceHandler extends Handler {
    public ServiceHandler(Looper looper) { super(looper); }

    public void handleMessage(Message msg) {
      onHandleIntent((Intent) msg.obj);

      stopSelf(msg.arg1);
    }

  }
  ...
```

Receives Messages passed via sendMessage()

# Implementation of IntentService

```
public class IntentService extends Service {
  ...

  private final class ServiceHandler extends Handler {
    public ServiceHandler(Looper looper) { super(looper); }

    public void handleMessage(Message msg) {
      onHandleIntent((Intent) msg.obj);

      stopSelf(msg.arg1);
    }

  }
  ...
```

Dispatch a callback hook method to process the Intent concurrently

# Implementation of IntentService

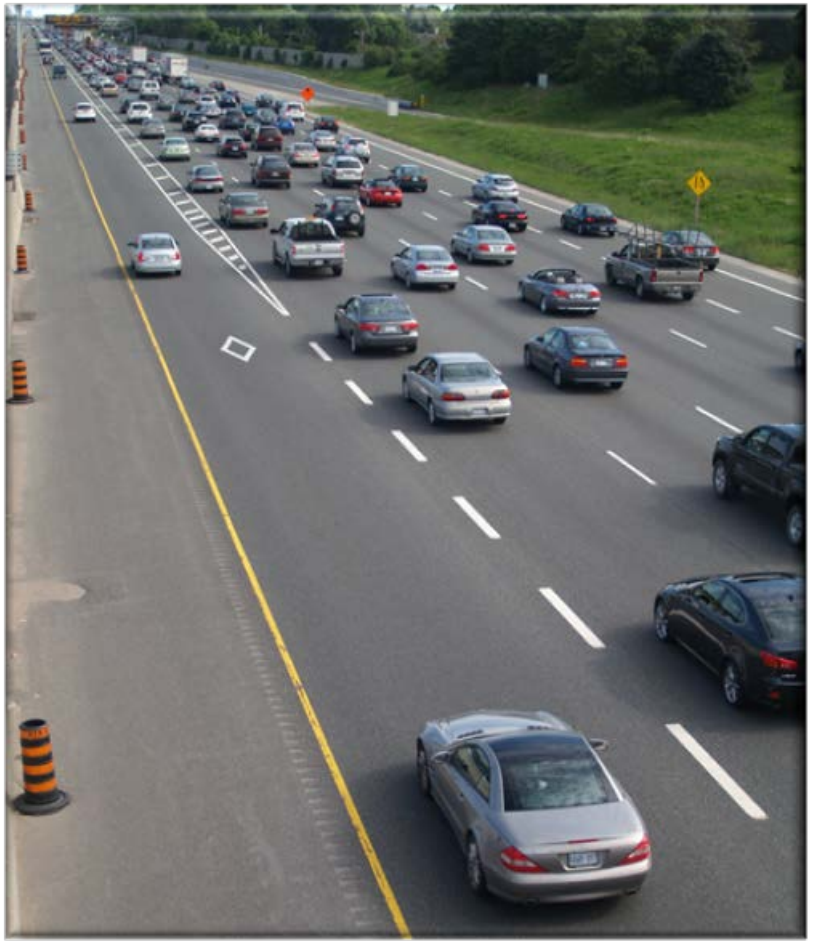
```
public class IntentService extends Service {
  ...

  private final class ServiceHandler extends Handler {
    public ServiceHandler(Looper looper) { super(looper); }

    public void handleMessage(Message msg) {
      onHandleIntent((Intent) msg.obj);

      stopSelf(msg.arg1);
    }

  }
  ...
```

Dispatch a callback hook method to process the Intent concurrently

# Implementation of IntentService

```
public class IntentService extends Service {
  ...
  protected abstract void
    onHandleIntent(Intent intent);

  ...
```

**Must be overridden by subclasses to process the Intent concurrently**



Only one Intent at a time is processed concurrently

# Implementation of IntentService

```
public class IntentService extends Service {
  ...

  private final class ServiceHandler extends Handler {
    public ServiceHandler(Looper looper) { super(looper); }

    public void handleMessage(Message msg) {
      onHandleIntent((Intent) msg.obj);

      stopSelf(msg.arg1);
    }
  }
  ...
```

**Stop the service using the startId, so that we don't stop the service in the middle of handling another job**

See previous part on “Programming Started Services”

# Implementation of IntentService

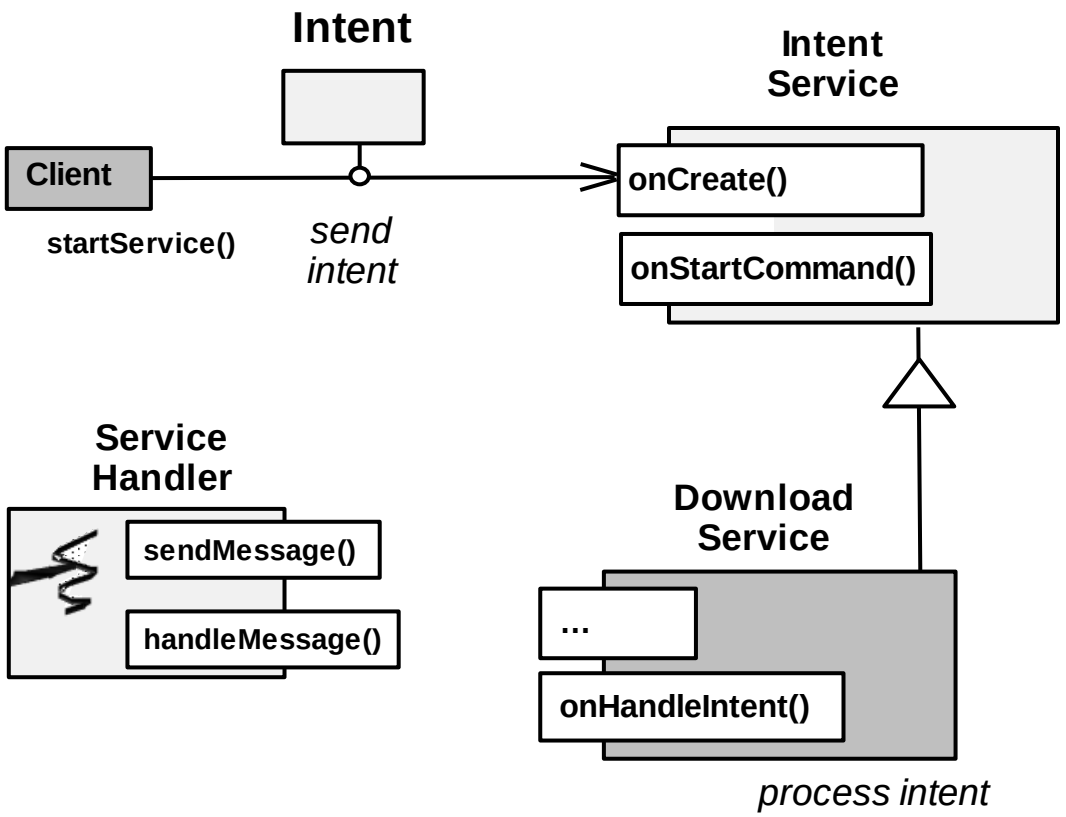
```
public class IntentService extends Service {
  ...

  public void onDestroy() {
    mServiceLooper.quit();
  }
}
```

Shutdown the looper

# Summary

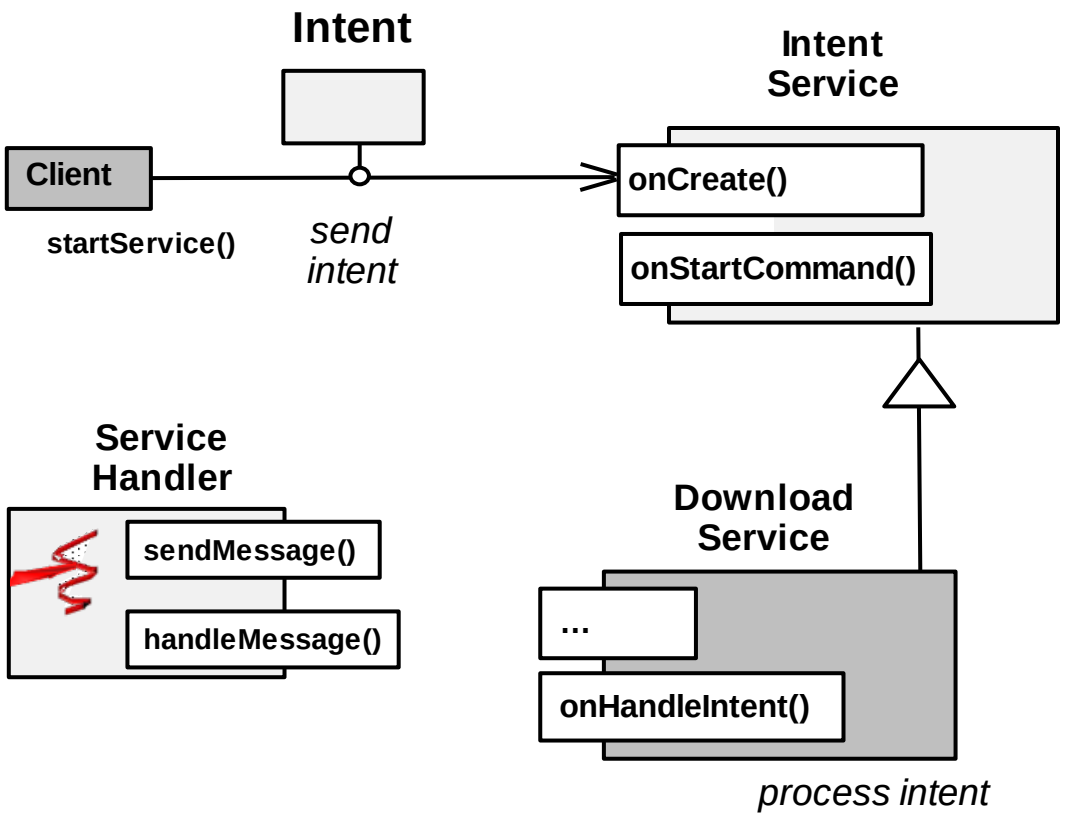
# Summary

- IntentService provides a framework that codifies a common Android idiom



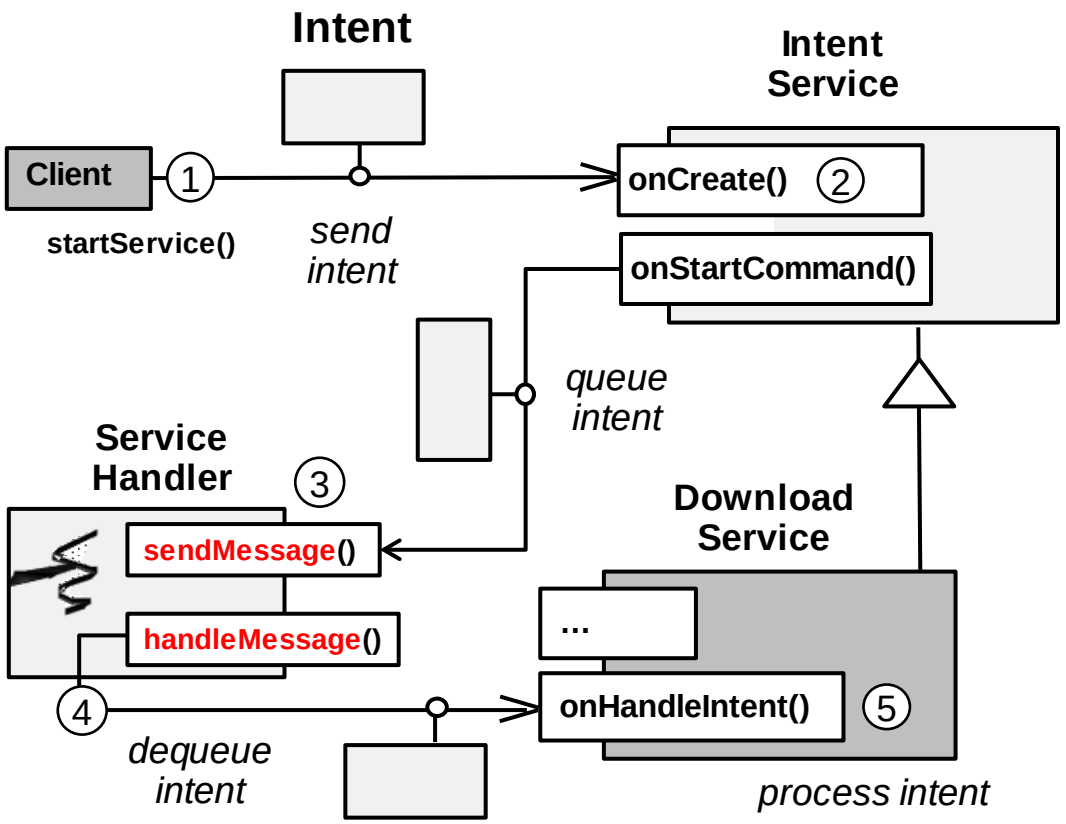
# Summary

- IntentService provides a framework that codifies a common Android idiom
  - It creates a Handler Thread that processes Intent commands in the background



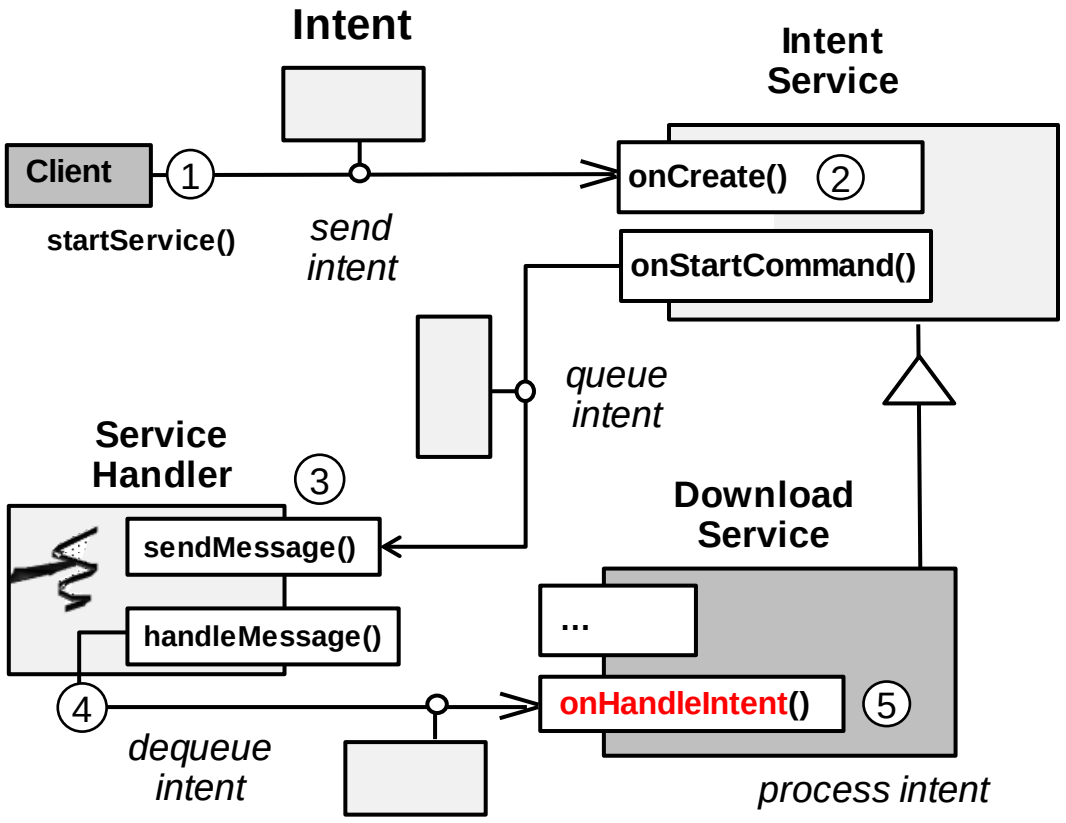
# Summary

- IntentService provides a framework that codifies a common Android idiom
  - It creates a Handler Thread that processes Intent commands in the background
  - It also uses the HaMeR framework to dispatch calls to onHandleIntent()



# Summary

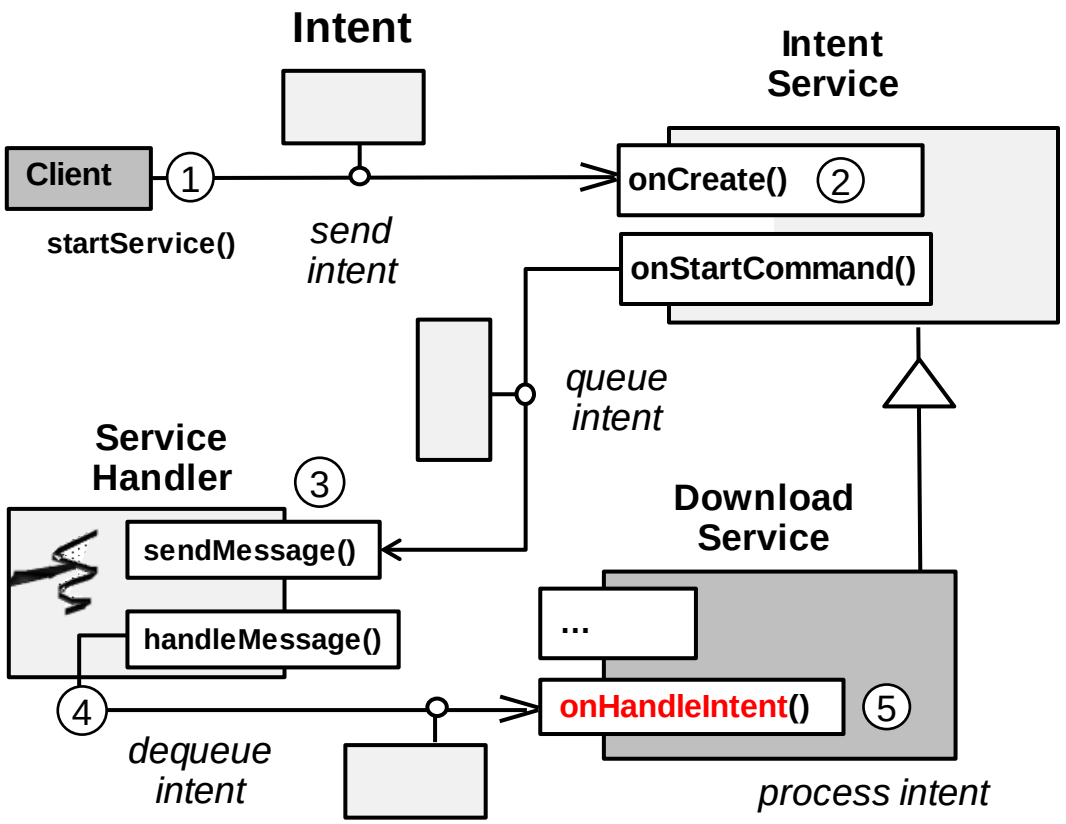
- IntentService provides a framework that codifies a common Android idiom
  - It creates a Handler Thread that processes Intent commands in the background
  - It also uses the HaMeR framework to dispatch calls to onHandleIntent()




IntentService is designed for Services that process one Intent request at a time

# Summary

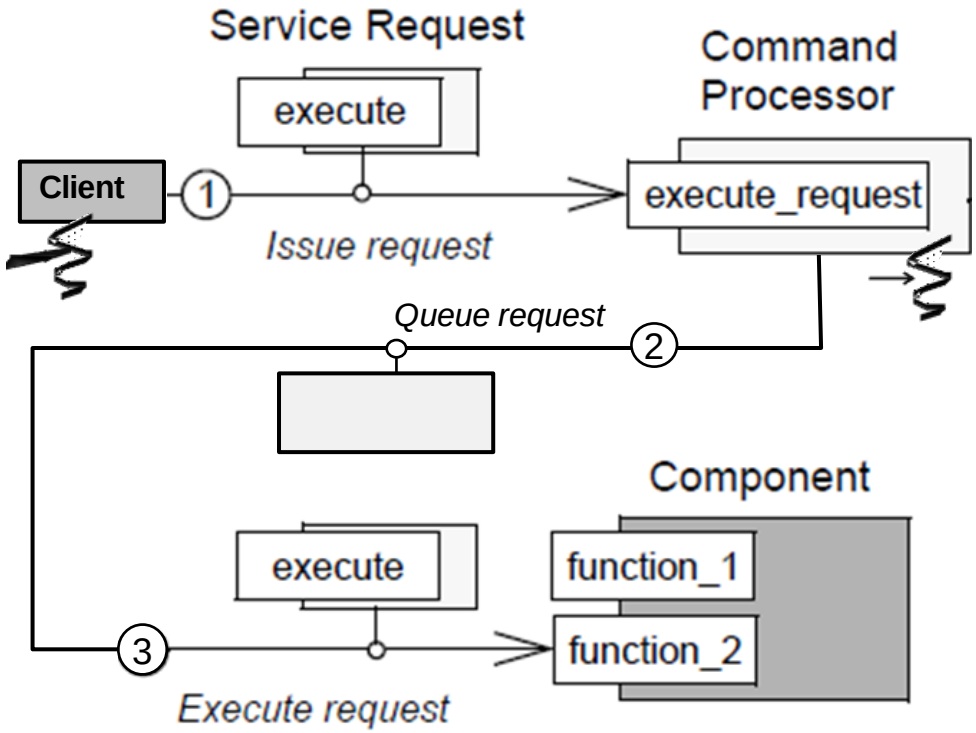
- IntentService provides a framework that codifies a common Android idiom
  - It creates a Handler Thread that processes Intent commands in the background
  - It also uses the HaMeR framework to dispatch calls to onHandleIntent()




Executors can be used to process Intents concurrently in a pool of Threads

# Summary

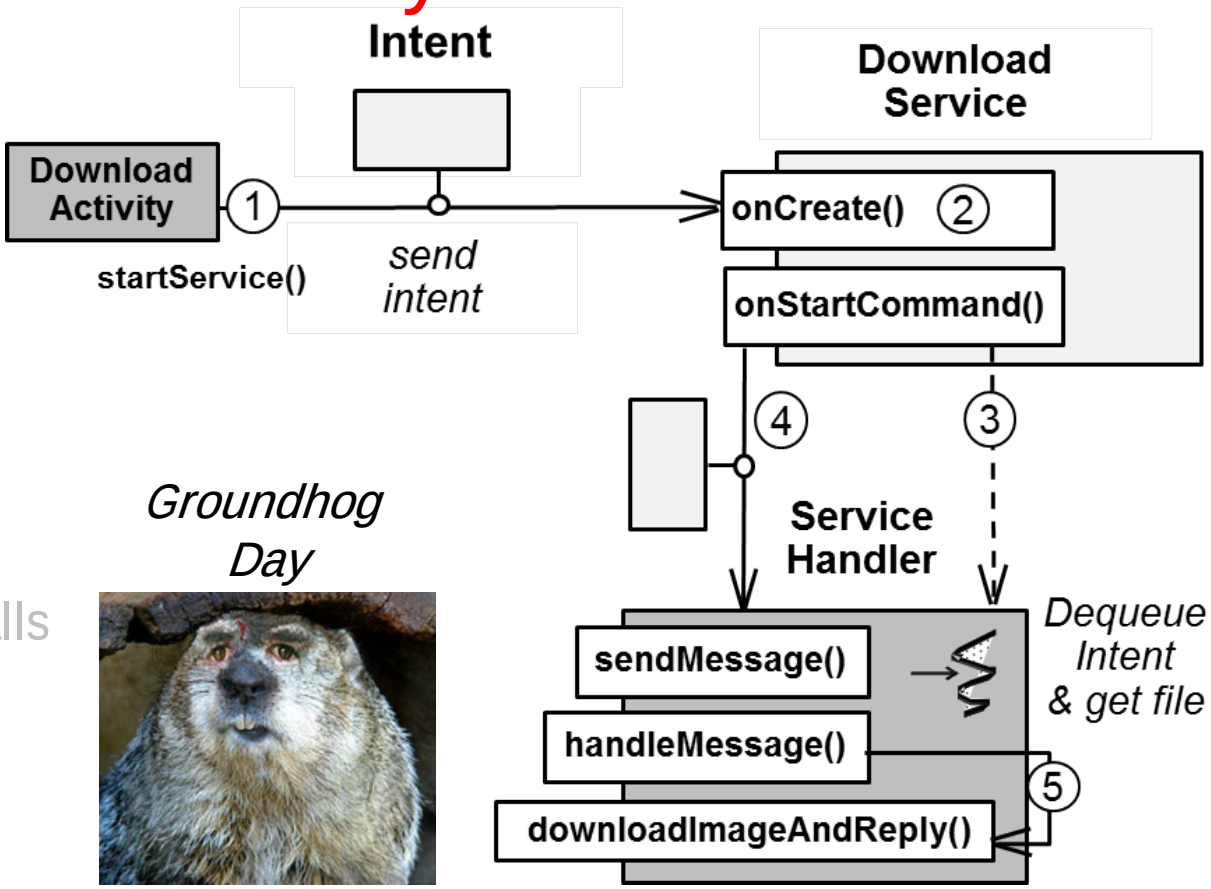
- IntentService provides a framework that codifies a common Android idiom
  - It creates a Handler Thread that processes Intent commands in the background
  - It also uses the HaMeR framework to serialize calls to onHandleIntent()
  - It applies the *Command Processor* pattern




See upcoming parts on “The Command Processor Pattern”

# Summary

- IntentService provides a framework that codifies a common Android idiom
  - It creates a Handler Thread that processes Intent commands in the background
  - It also uses the HaMeR framework to serialize calls to onHandleIntent()
  - It applies the *Command Processor* pattern
  - It shows how frameworks evolve organically by refactoring software




*Groundhog Day*



See previous parts on “Programming Started Services”

# Summary

- IntentService provides a framework that codifies a common Android idiom
- IntentService is heavily used in Android's packaged applications

packages/apps/Calendar/src/com/android/calendar/alerts/
    DismissAllAlarmsService.java
packages/apps/CellBroadcastReceiver/src/com/android/cellbroadcastreceiver/
    CellBroadcastDatabaseService.java
packages/apps/Contacts/src/com/android/contacts/ContactSaveService.java
packages/apps/Email/src/com/android/email/service/
    EmailBroadcastProcessorService.java
packages/apps/Exchange/src/com/android/exchange/service/
    ExchangeBroadcastProcessorService.java
packages/apps/MusicFX/src/com/android/musicfx/Compatibility.java
packages/apps/Phone/src/com/android/phone/ClearMissedCallsService.java

# Summary

- IntentService provides a framework that codifies a common Android idiom
- IntentService is heavily used in Android's packaged applications

packages/apps/Calendar/src/com/android/calendar/alerts/
DismissAllAlarmsService.java
packages/apps/CellBroadcastReceiver/src/com/android/cellbroadcastreceiver/
CellBroadcastDatabaseService.java
packages/apps/Contacts/src/com/android/contacts/ContactSaveService.java
packages/apps/Email/src/com/android/email/service/
EmailBroadcastProcessorService.java
packages/apps/Exchange/src/com/android/exchange/service/
ExchangeBroadcastProcessorService.java
packages/apps/MusicFX/src/com/android/musicfx/Compatibility.java
packages/apps/Phone/src/com/android/phone/ClearMissedCallsService.java

# Summary

- IntentService provides a framework that codifies a common Android idiom
- IntentService is heavily used in Android's packaged applications

packages/apps/Calendar/src/com/android/calendar/alerts/
    DismissAllAlarmsService.java
packages/apps/CellBroadcastReceiver/src/com/android/cellbroadcastreceiver/
    CellBroadcastDatabaseService.java
packages/apps/Contacts/src/com/android/contacts/ContactSaveService.java
packages/apps/Email/src/com/android/email/service/
    EmailBroadcastProcessorService.java
packages/apps/Exchange/src/com/android/exchange/service/
    ExchangeBroadcastProcessorService.java
packages/apps/MusicFX/src/com/android/musicfx/Compatibility.java
packages/apps/Phone/src/com/android/phone/ClearMissedCallsService.java