Java 8 Parallel ImageStreamGang Example (Part 2)

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

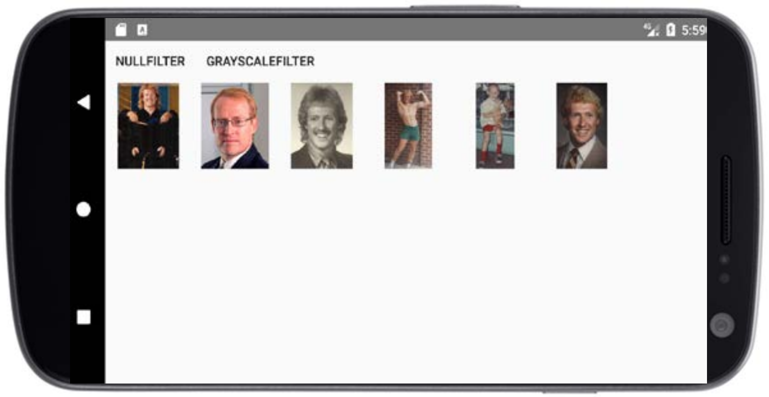
Professor of Computer Science

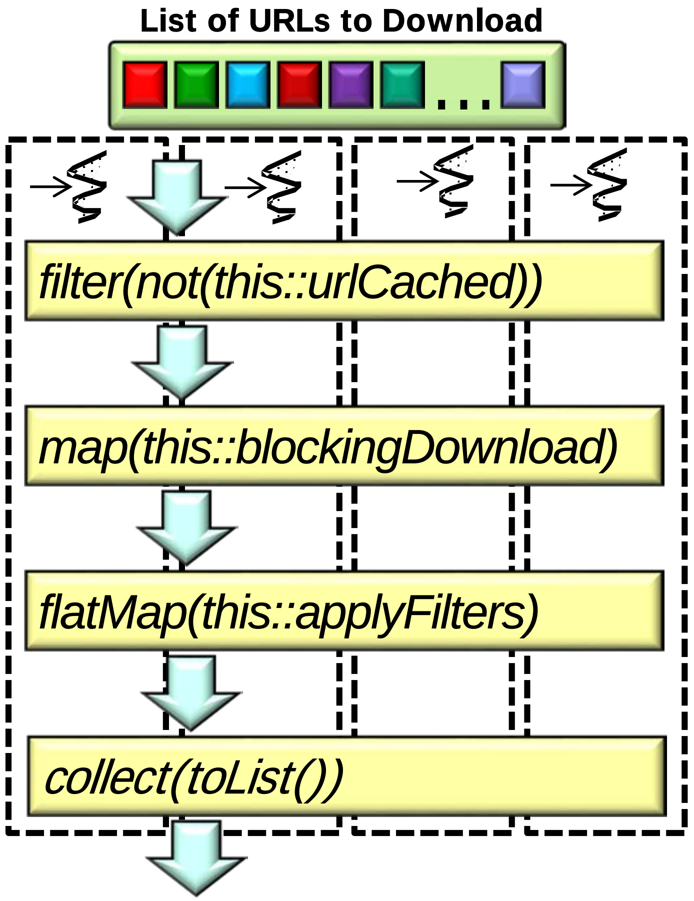
Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of an ImageStreamGang app
- Know how Java 8 parallel streams are applied to the ImageStreamGang app

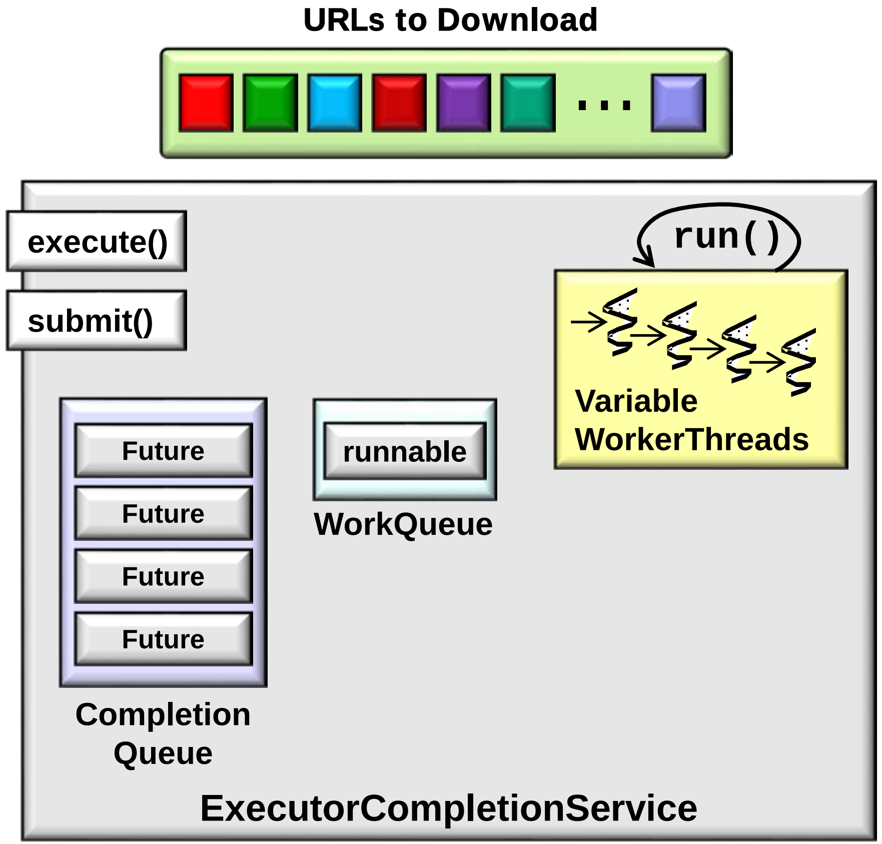



List of URLs to Download
filter(not(this::urlCached))
map(this::blockingDownload)
flatMap(this::applyFilters)
collect(toList())

See github.com/douglascraigschmidt/LiveLessons/blob/master/ImageStreamGang

Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of an ImageStreamGang app
- Know how Java 8 parallel streams are applied to the ImageStreamGang app
  - This app enhances ImageTaskGang

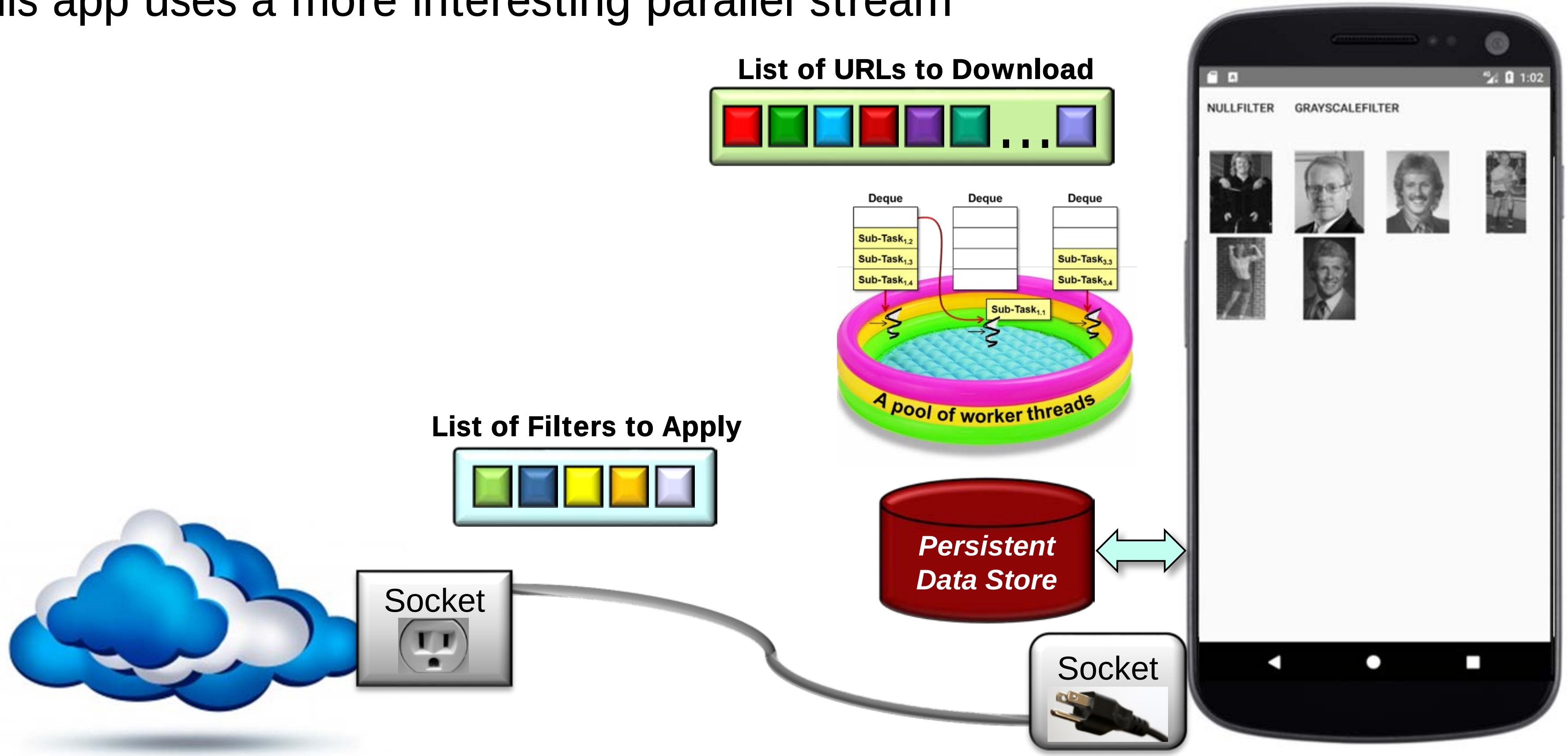


See github.com/douglascraigschmidt/LiveLessons/tree/master/ImageTaskGangApplication

Overview of Parallel Streams in ImageStreamGang

Overview of Parallel Streams in ImageStreamGang

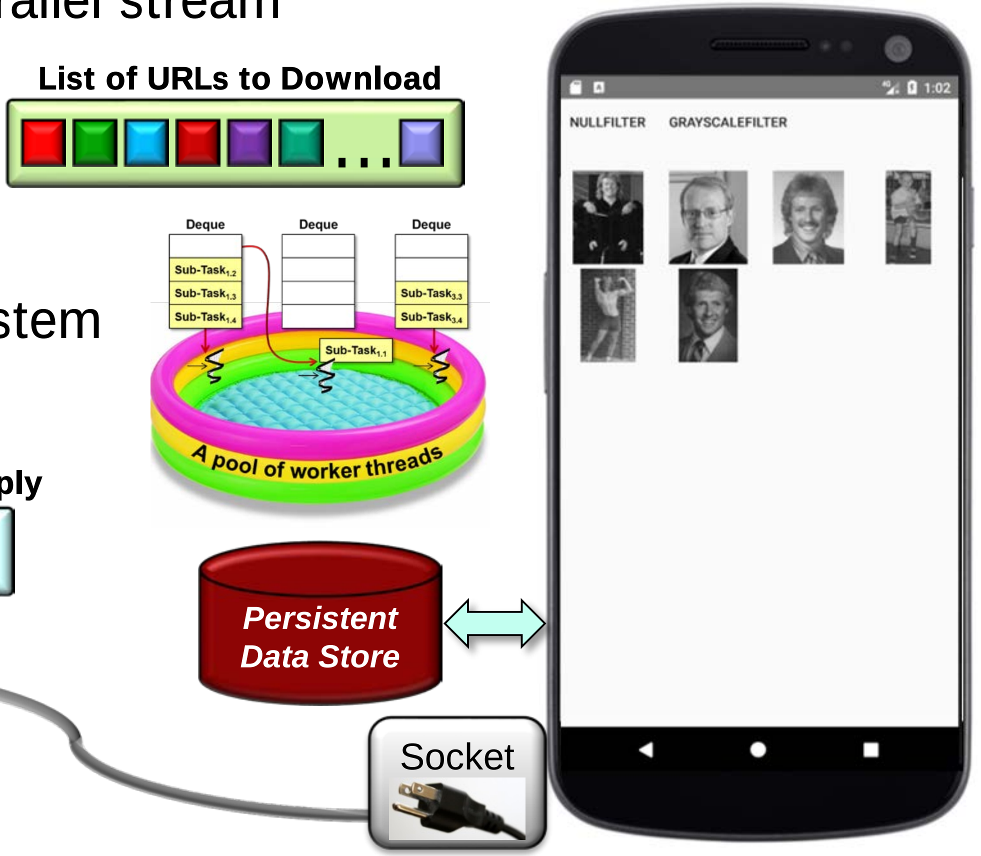
- This app uses a more interesting parallel stream

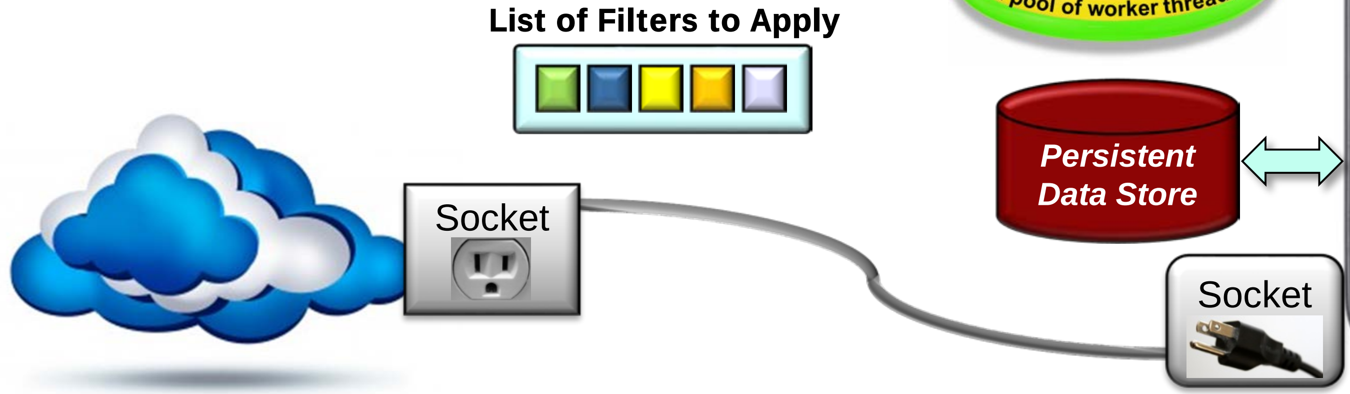


See github.com/douglascraigschmidt/LiveLessons/blob/master/ImageStreamGang/AndroidGUI

Overview of Parallel Streams in ImageStreamGang

- This app uses a more interesting parallel stream
  - Ignore cached images
  - Download non-cached images
  - Apply list of filters to each image
  - Store filtered images in the file system
  - Display images to the user

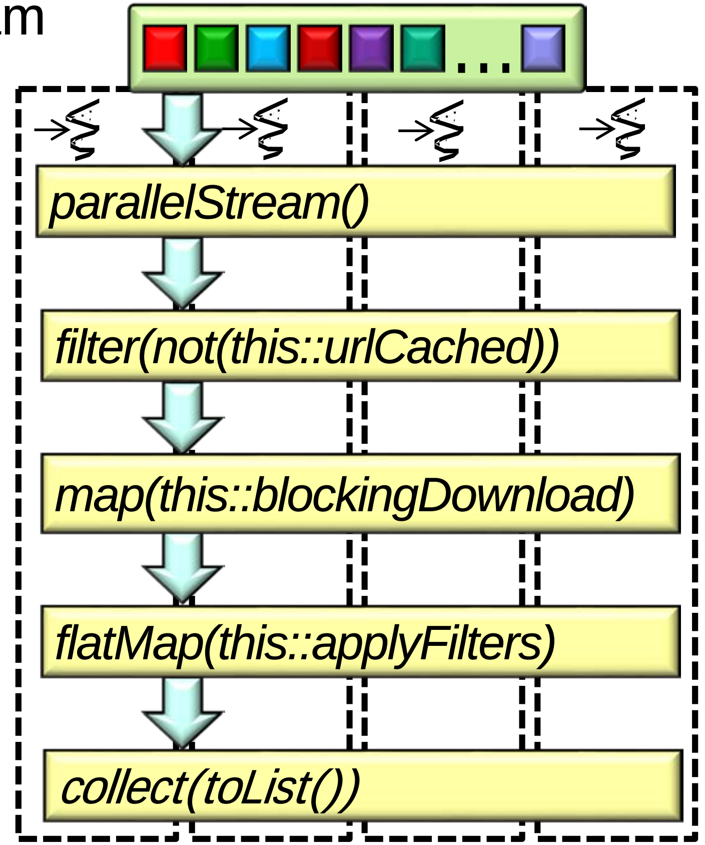




Combines Java 8 object-oriented & functional programming features

Overview of Parallel Streams in ImageStreamGang

- This app uses a more interesting parallel stream
  - Ignore cached images
  - Download non-cached images
  - Apply list of filters to each image
  - Store filtered images in the file system
  - Display images to the user (after triggering stream processing)

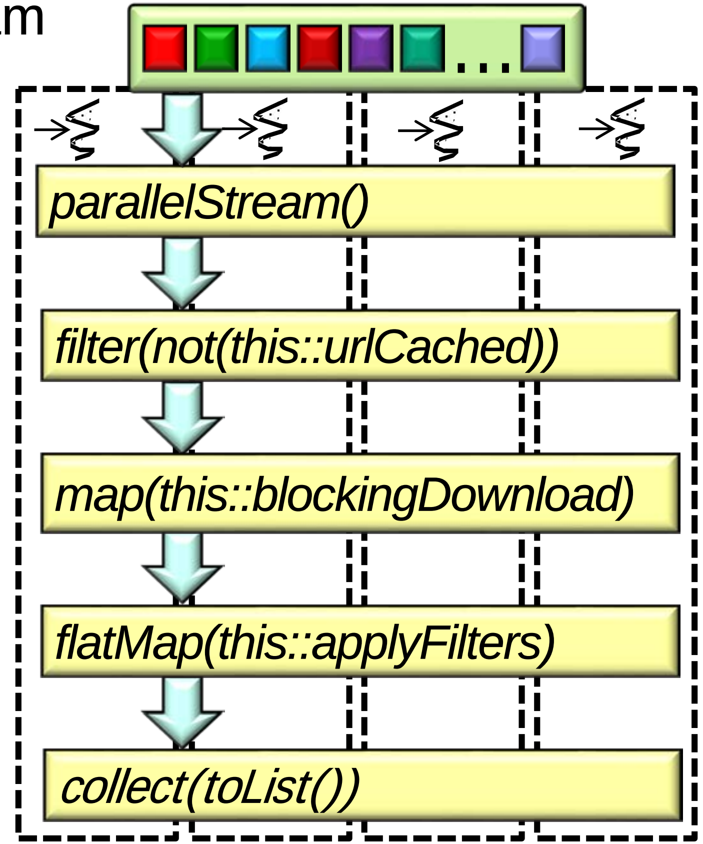


...
parallelStream()
filter(not(this::urlCached))
map(this::blockingDownload)
flatMap(this::applyFilters)
collect(toList())

Declarative stream pipeline closely aligns with the app description

Overview of Parallel Streams in ImageStreamGang

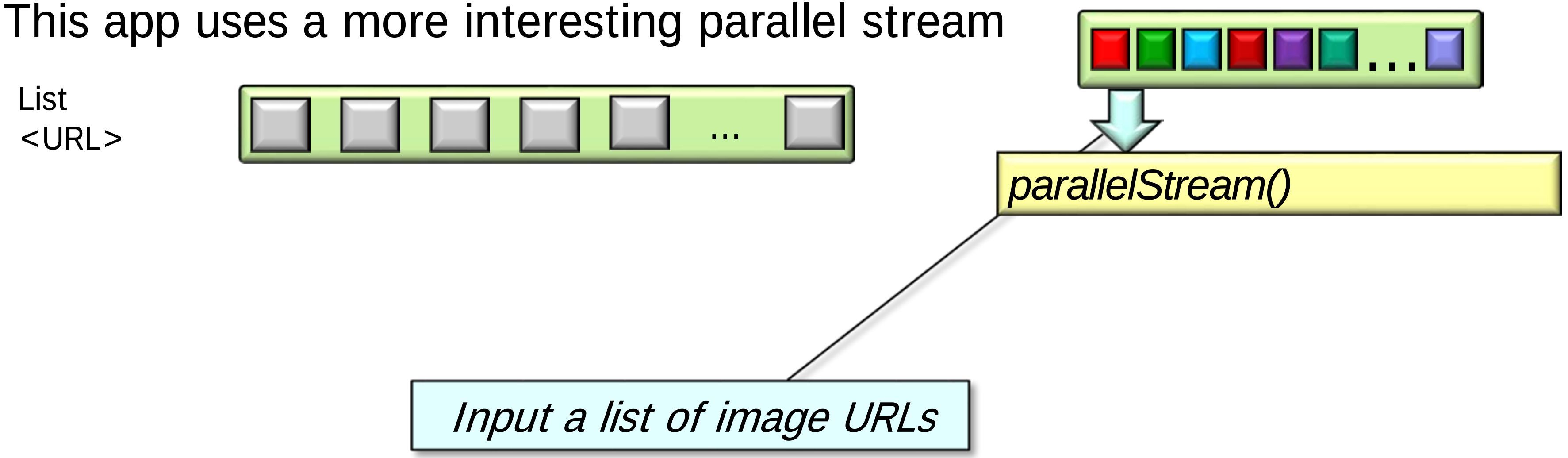
- This app uses a more interesting parallel stream


parallelStream()
filter(not(this::urlCached))
map(this::blockingDownload)
flatMap(this::applyFilters)
collect(toList())

Closes gap between design intent & computations that implement the intent

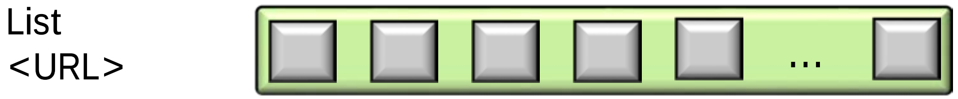
Overview of Parallel Streams in ImageStreamGang

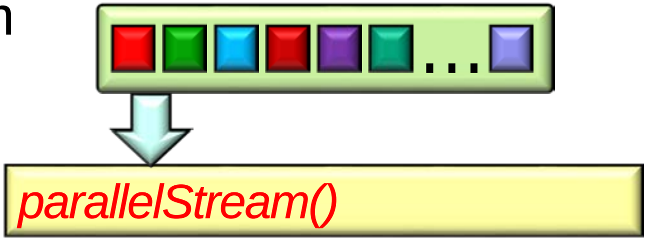
- This app uses a more interesting parallel stream



Overview of Parallel Streams in ImageStreamGang

- This app uses a more interesting parallel stream

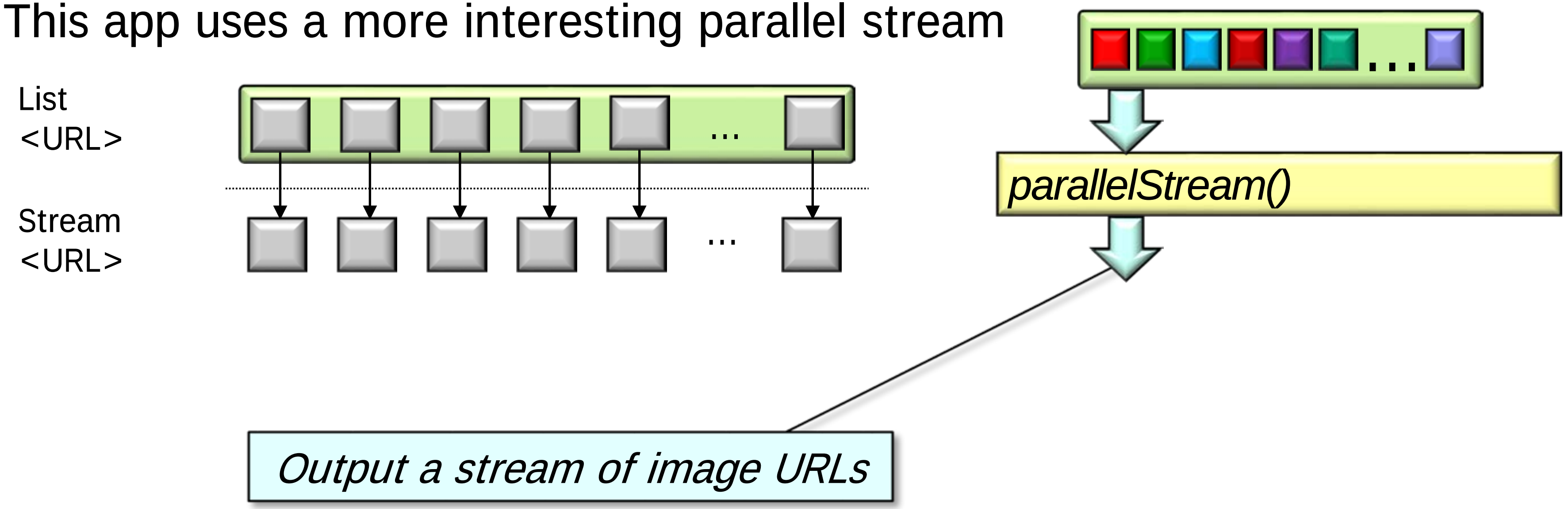




Convert collection to a parallel stream

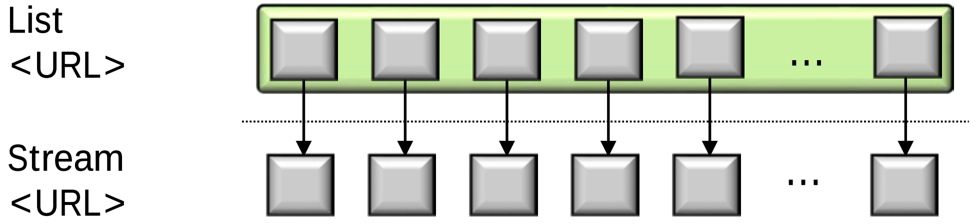
Overview of Parallel Streams in ImageStreamGang

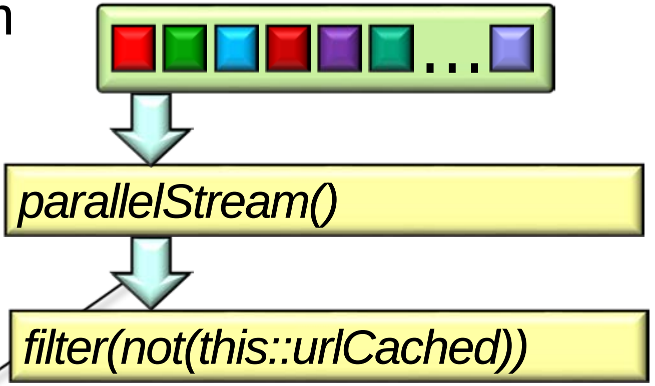
- This app uses a more interesting parallel stream



Overview of Parallel Streams in ImageStreamGang

- This app uses a more interesting parallel stream

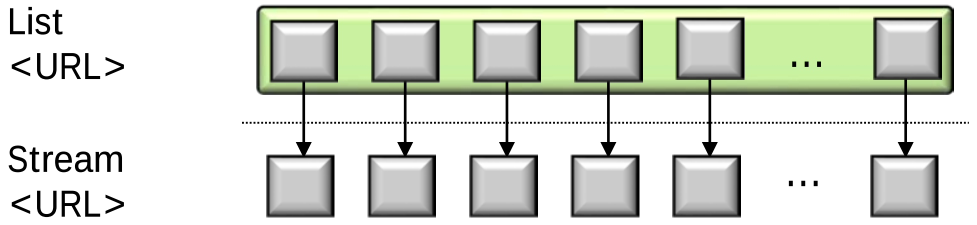


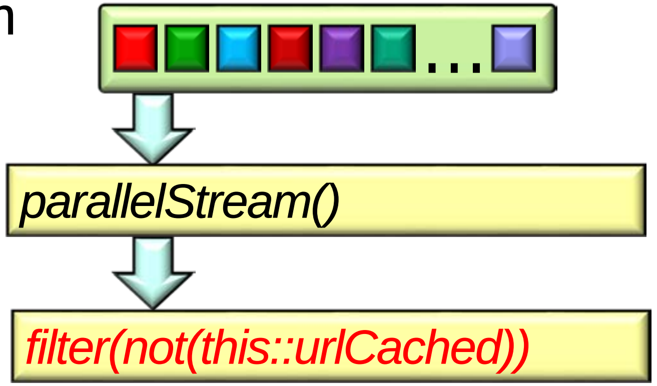


Input a stream of image URLs

Overview of Parallel Streams in ImageStreamGang

- This app uses a more interesting parallel stream

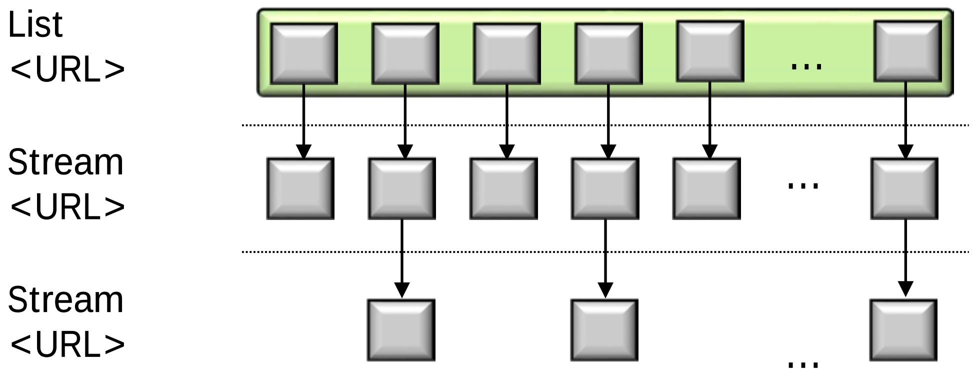


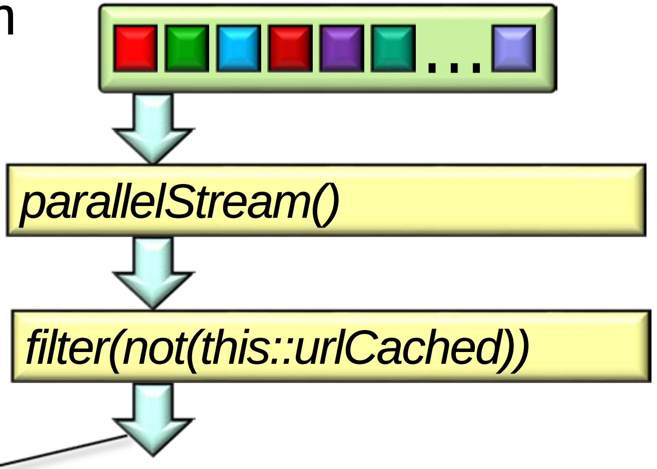


filter() ignores cached images

Overview of Parallel Streams in ImageStreamGang

- This app uses a more interesting parallel stream

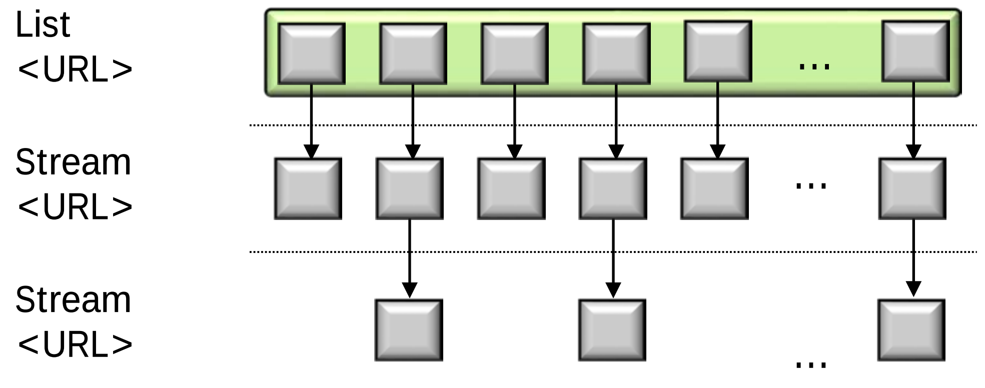


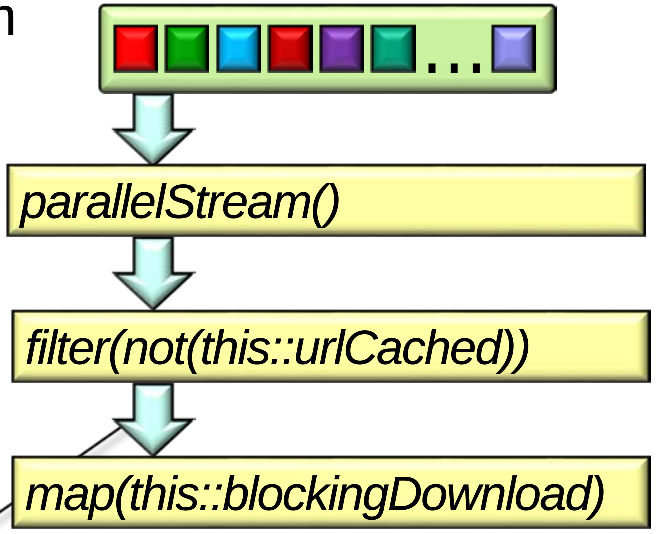


Output a stream of filtered image URLs

Overview of Parallel Streams in ImageStreamGang

- This app uses a more interesting parallel stream

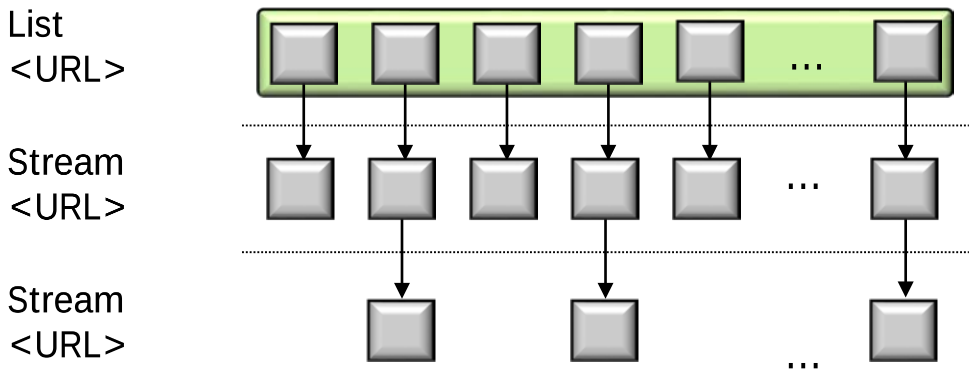


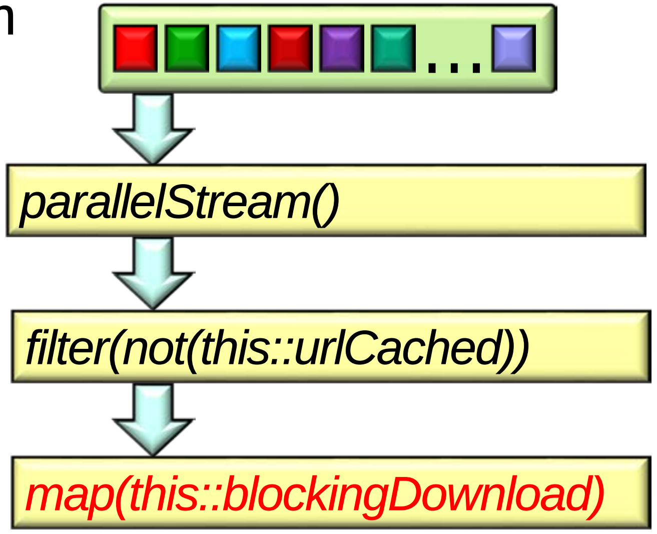


Input a stream of filtered image URLs

Overview of Parallel Streams in ImageStreamGang

- This app uses a more interesting parallel stream

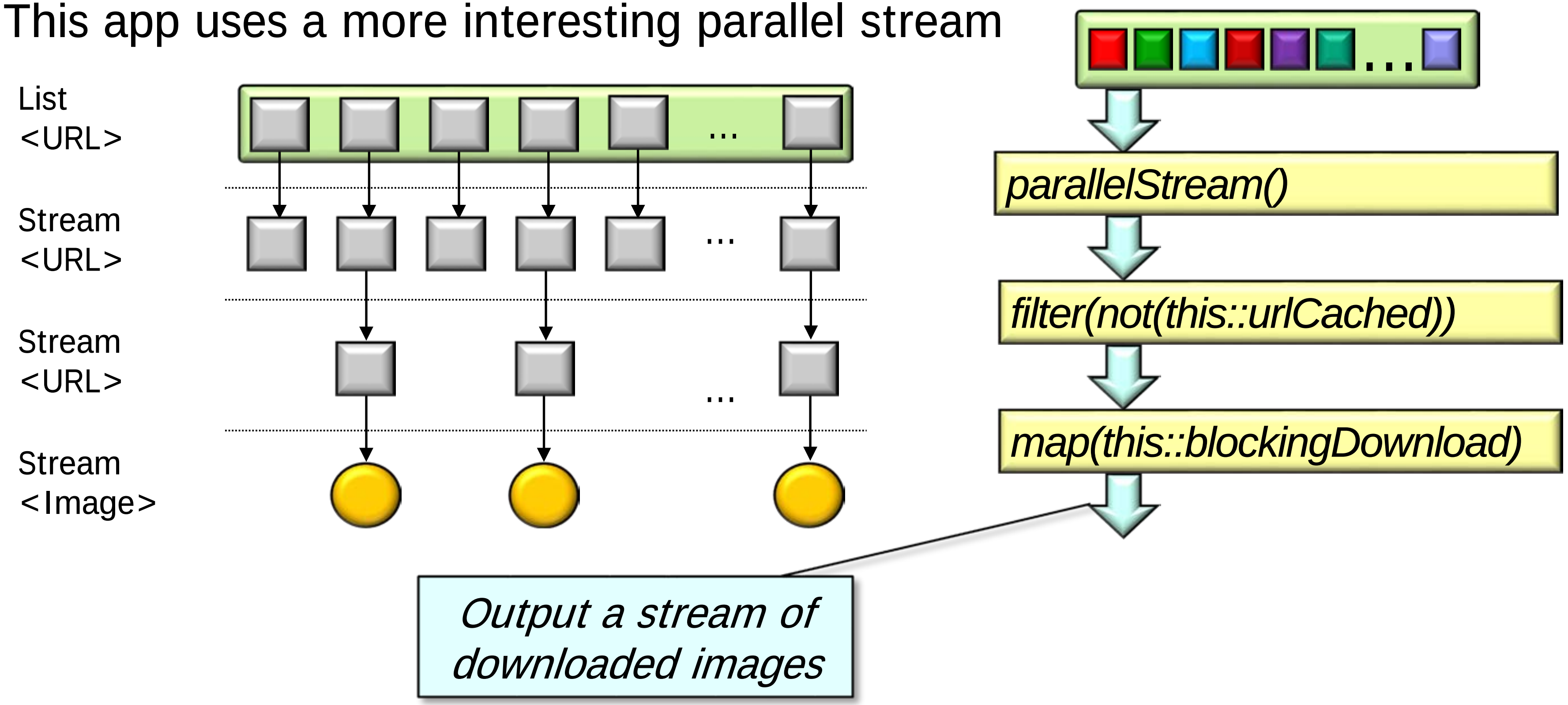




Download non-cached images

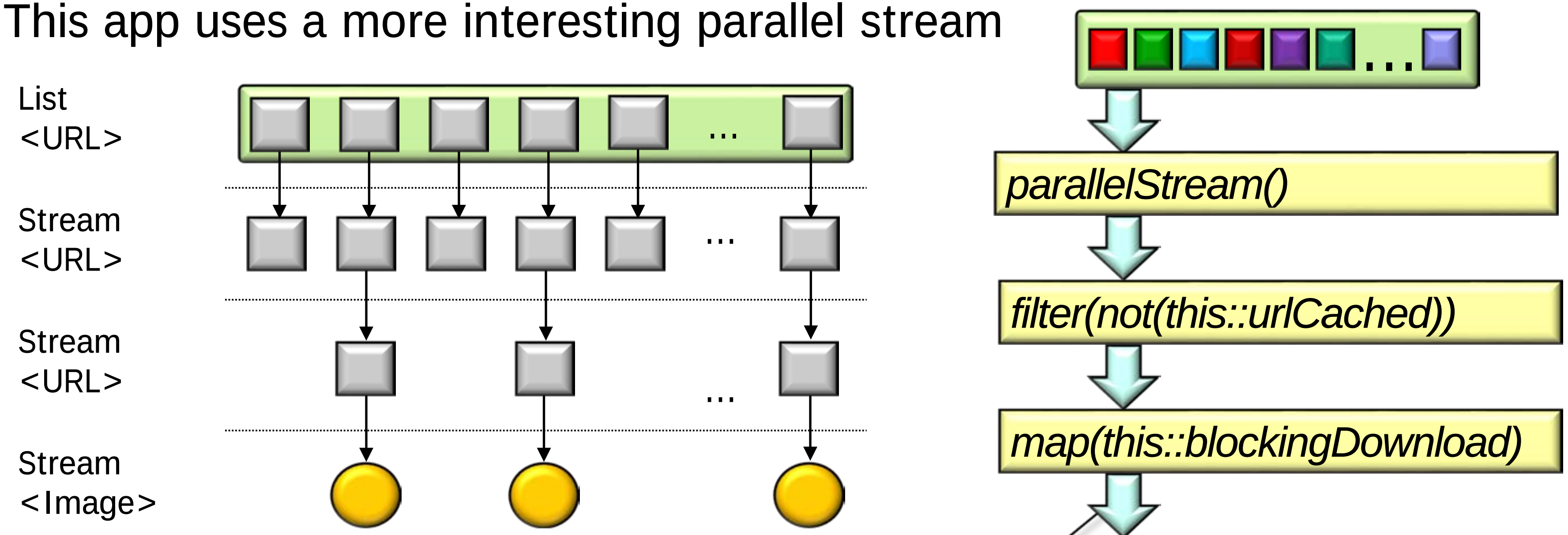
Overview of Parallel Streams in ImageStreamGang

- This app uses a more interesting parallel stream


List <URL>
Stream <URL>
Stream <URL>
Stream <Image>
...
parallelStream()
filter(not(this::urlCached))
map(this::blockingDownload)
Output a stream of downloaded images

Overview of Parallel Streams in ImageStreamGang

- This app uses a more interesting parallel stream

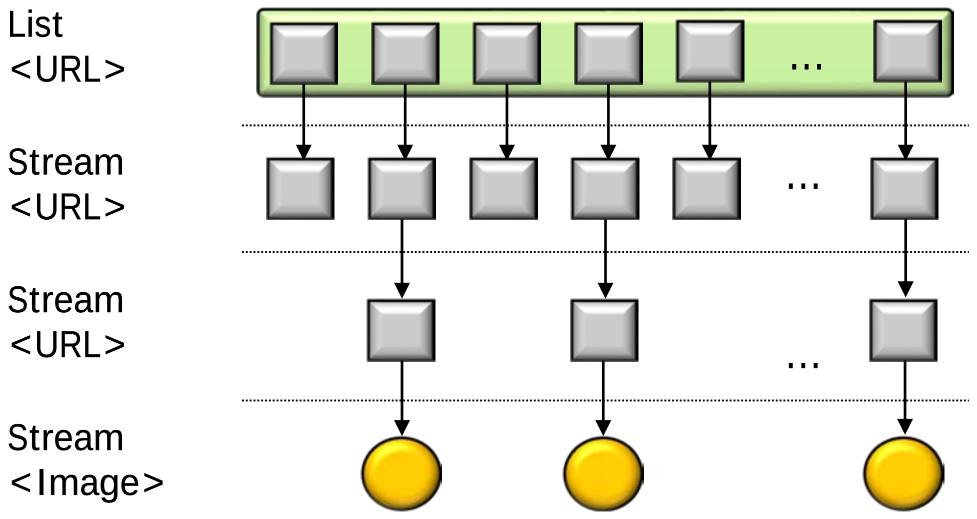


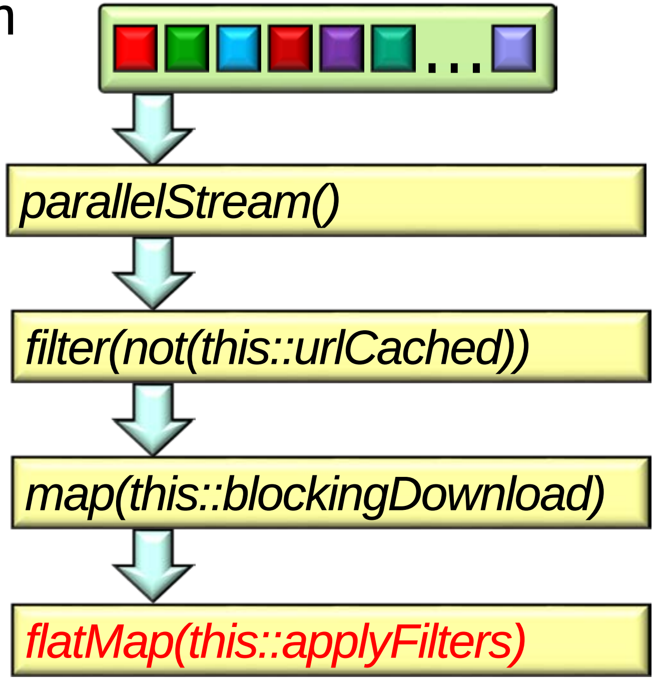
List <URL>
Stream <URL>
Stream <URL>
Stream <Image>
...
parallelStream()
filter(not(this::urlCached))
map(this::blockingDownload)
flatMap(this::applyFilters)

Input a stream of downloaded images

Overview of Parallel Streams in ImageStreamGang

- This app uses a more interesting parallel stream

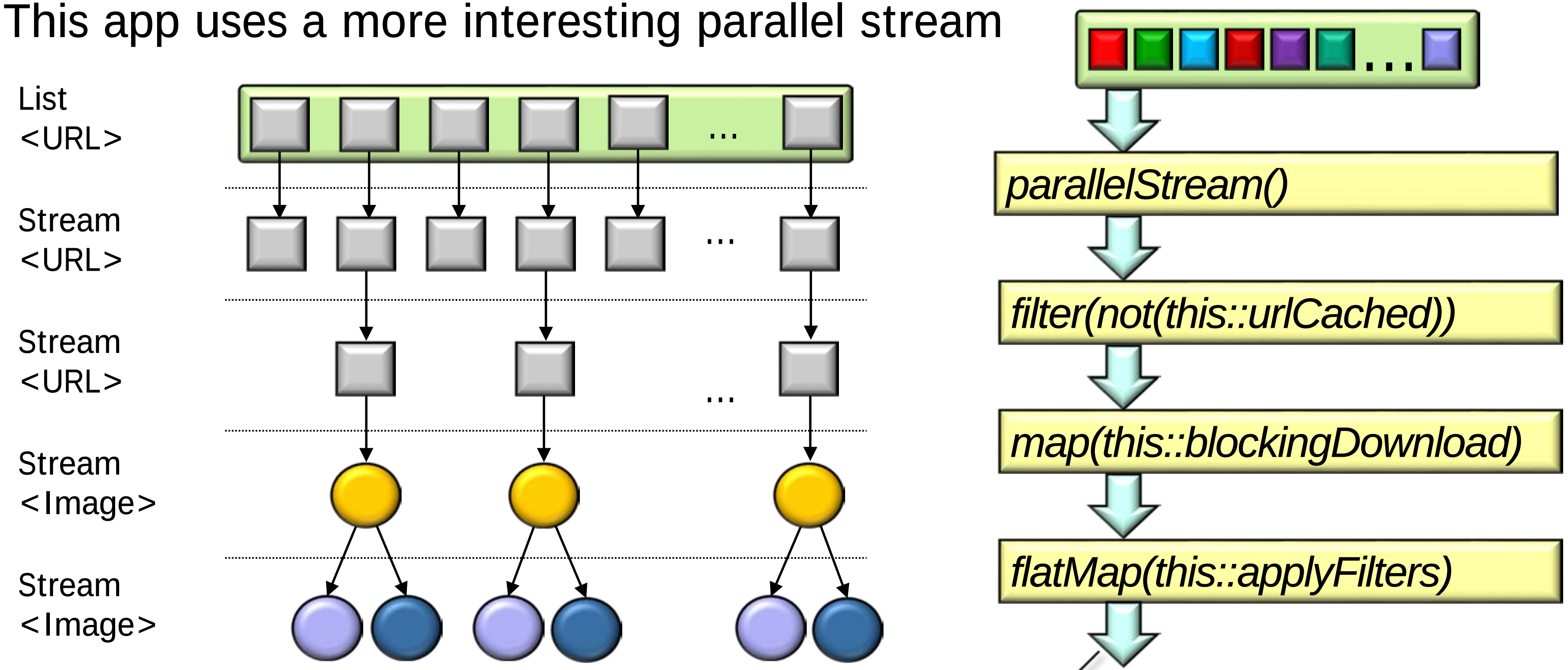



...
parallelStream()
filter(not(this::urlCached))
map(this::blockingDownload)
flatMap(this::applyFilters)

Apply list of filters to each image & store filtered images in the file system

Overview of Parallel Streams in ImageStreamGang

- This app uses a more interesting parallel stream

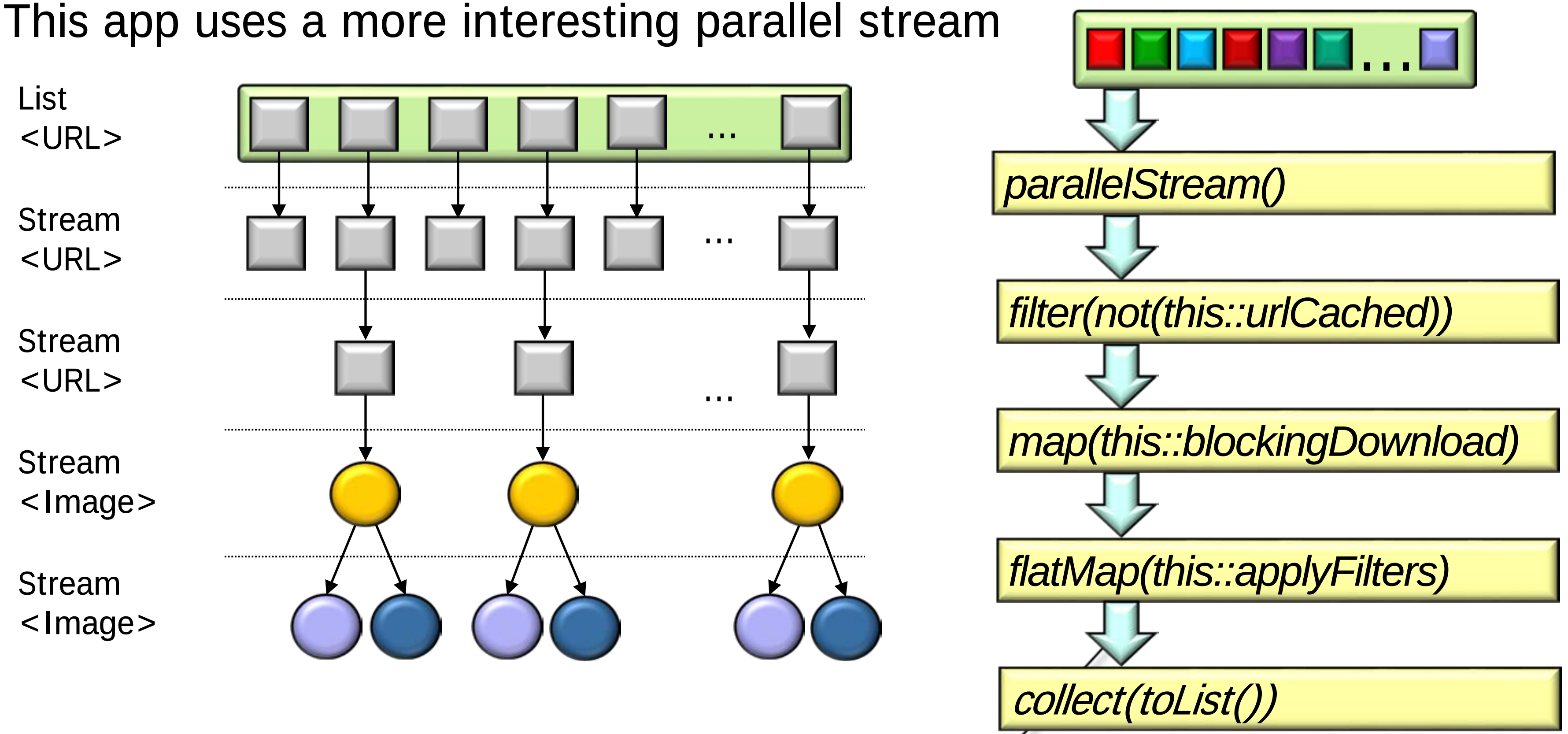


List <URL>
Stream <URL>
Stream <URL>
Stream <Image>
Stream <Image>
parallelStream()
filter(not(this::urlCached))
map(this::blockingDownload)
flatMap(this::applyFilters)

Output a stream of filtered & stored images

Overview of Parallel Streams in ImageStreamGang

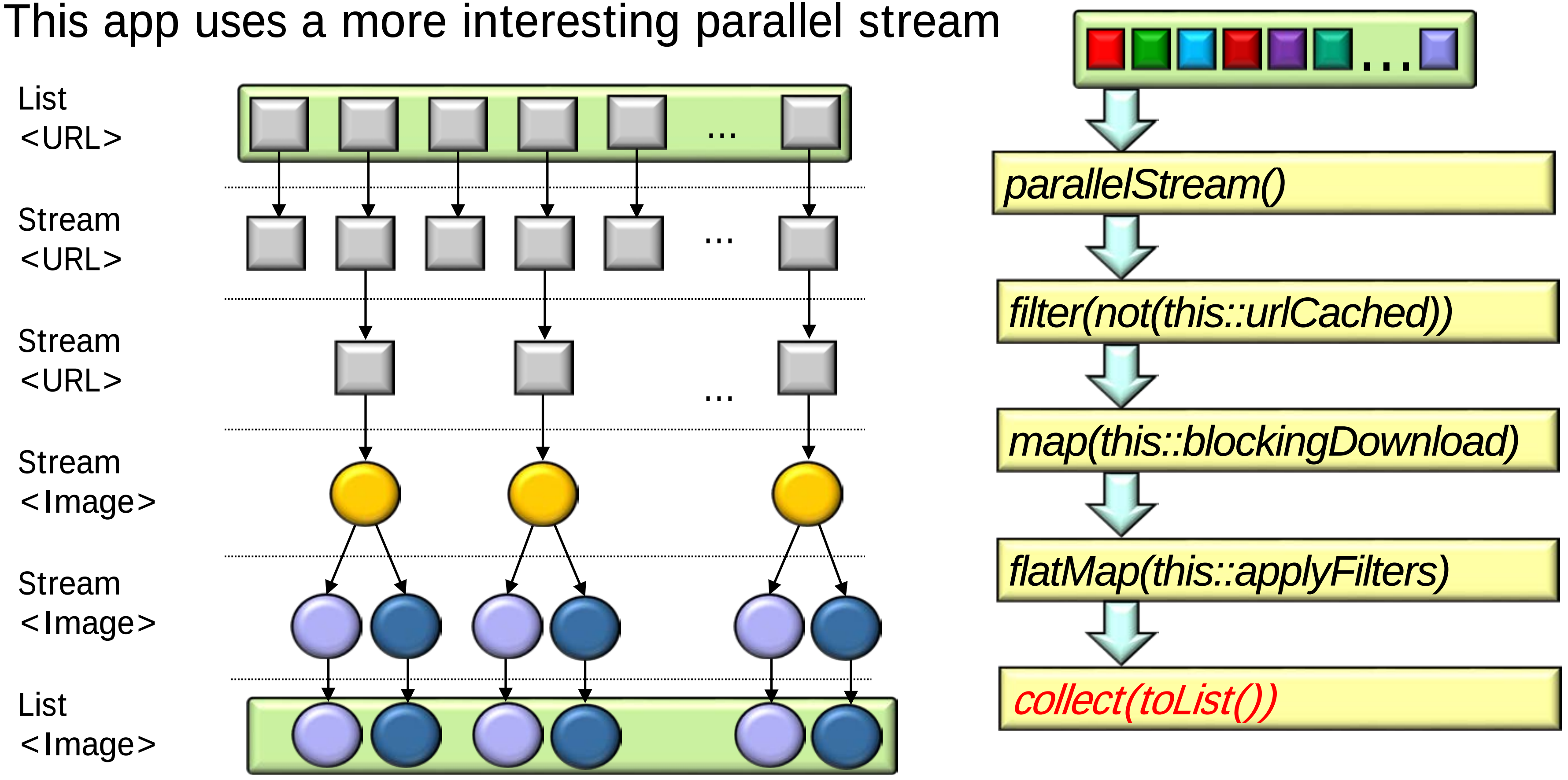
- This app uses a more interesting parallel stream



Input a stream of filtered & stored images

Overview of Parallel Streams in ImageStreamGang

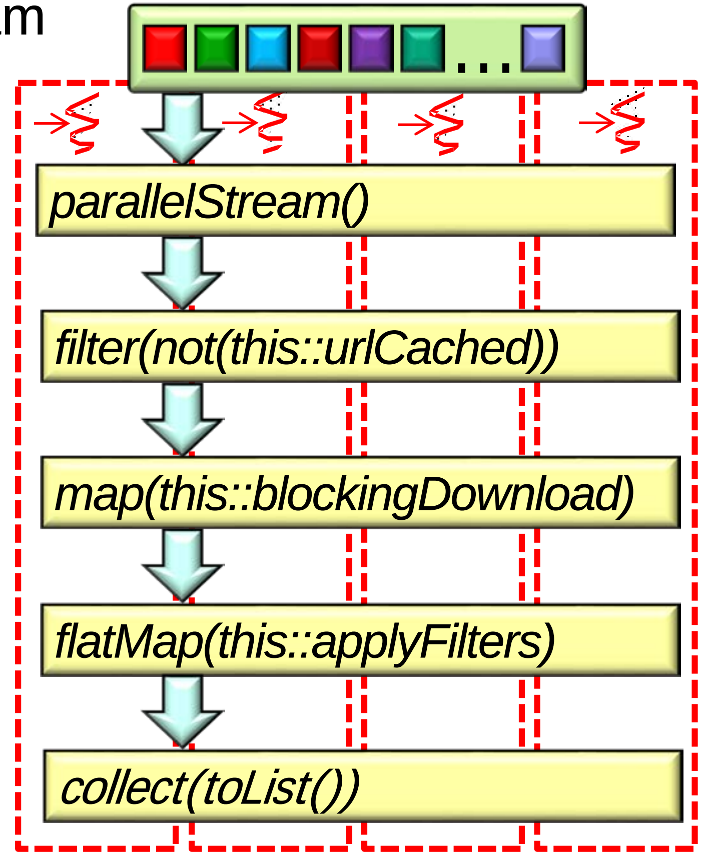
- This app uses a more interesting parallel stream



Trigger stream processing & return a list of filtered & stored images

Overview of Parallel Streams in ImageStreamGang

- This app uses a more interesting parallel stream
  - Ignore cached images
  - Download non-cached images
  - Apply list of filters to each image
  - Store filtered images in the file system
  - Display images to the user (after triggering stream processing)


...
parallelStream()
filter(not(this::urlCached))
map(this::blockingDownload)
flatMap(this::applyFilters)
collect(toList())

The Java 8 streams framework orchestrates all these steps in parallel

Applying Parallel Streams to ImageStreamGang

Implentating a Parallel Stream in ImageStreamGang

- We focus on processStream() in ImageStreamParallel.java

```
void processStream() {
  List<URL> urls = getInput();

  List<Image> filteredImages = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::blockingDownload)
    .flatMap(this::applyFilters)
    .collect(toList());

  System.out.println(TAG
        + "Image(s) filtered = "
        + filteredImages.size());
}
```

See app/src/main/java/livelessons/imagestreamgang/streams/ImageStreamParallel.java

Implentating a Parallel Stream in ImageStreamGang

- We focus on processStream() in ImageStreamParallel.java

Get a list of URLs

```
void processStream() {
  List<URL> urls = getInput();

  List<Image> filteredImages = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::blockingDownload)
    .flatMap(this::applyFilters)
    .collect(toList());

  System.out.println(TAG
        + "Image(s) filtered = "
        + filteredImages.size());
}
```

Implentating a Parallel Stream in ImageStreamGang

- We focus on processStream() in ImageStreamParallel.java

```
void processStream() {
  List<URL> urls = getInput();

  List<Image> filteredImages = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::blockingDownload)
    .flatMap(this::applyFilters)
    .collect(toList());

  System.out.println(TAG
          + "Image(s) filtered = "
          + filteredImages.size());
}
```

Convert a collection into a parallel stream

Implentating a Parallel Stream in ImageStreamGang

- We focus on processStream() in ImageStreamParallel.java

```
void processStream() {
  List<URL> urls = getInput();

  List<Image> filteredImages = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::blockingDownload)
    .flatMap(this::applyFilters)
    .collect(toList());

  System.out.println(TAG
        + "Image(s) filtered = "
        + filteredImages.size());
}
```

Return an output stream consisting of the URLs in the input stream that are not already cached

See docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#filter

Implentating a Parallel Stream in ImageStreamGang

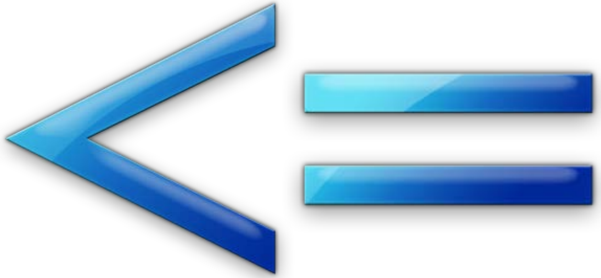
- We focus on processStream() in ImageStreamParallel.java

```
void processStream() {
  List<URL> urls = getInput();

  List<Image> filteredImages = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::blockingDownload)
    .flatMap(this::applyFilters)
    .collect(toList());

  System.out.println(TAG
        + "Image(s) filtered = "
        + filteredImages.size());
}
```

Return an output stream consisting of the URLs in the input stream that are not already cached



# of output stream elements will be <= # of input stream elements

Implentating a Parallel Stream in ImageStreamGang

- We focus on processStream() in ImageStreamParallel.java

```
void processStream() {
   List<URL> urls = getInput();

   List<Image> filteredImages = urls
      .parallelStream()
      .filter(not(this::urlCached))
      .map(this::blockingDownload)
      .flatMap(this::applyFilters)
      .collect(toList());

   System.out.println(TAG
         + "Image(s) filtered = "
         + filteredImages.size());
}
```

```
boolean urlCached(URL url) {
   return mFilters
     .stream()
     .filter(filter ->
        urlCached(url,
           filter.getName()))
     .count() > 0;
}
```

See app/src/main/java/livelessons/imagestreamgang/streams/ImageStreamGang.java

Implentating a Parallel Stream in ImageStreamGang

- We focus on processStream() in ImageStreamParallel.java

```
void processStream() {
  List<URL> urls = getInput();

  List<Image> filteredImages = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::blockingDownload)
    .flatMap(this::applyFilters)
    .collect(toList());

  System.out.println(TAG
        + "Image(s) filtered = "
        + filteredImages.size());
}
```

```
boolean urlCached(URL url,
         String filterName) {
  File file =
    new File(getPath(),
             filterName);

  File imageFile =
    new File(file,
         getNameForUrl(url));

  return imageFile.exists();
}
```

See app/src/main/java/livelessons/imagestreamgang/streams/ImageStreamGang.java

Implentating a Parallel Stream in ImageStreamGang

- We focus on processStream() in ImageStreamParallel.java

```
void processStream() {
  List<URL> urls = getInput();

  List<Image> filteredImages = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::blockingDownload)
    .flatMap(this::applyFilters)
    .collect(toList());

  System.out.println(TAG
        + "Image(s) filtered = "
        + filteredImages.size());
}
```

Return an output stream consisting of the images that were downloaded from the URLs in the input stream

See docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#map

Implentating a Parallel Stream in ImageStreamGang

- We focus on processStream() in ImageStreamParallel.java

```
void processStream() {
  List<URL> urls = getInput();

  List<Image> filteredImages = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::blockingDownload)
    .flatMap(this::applyFilters)
    .collect(toList());

  System.out.println(TAG
        + "Image(s) filtered = "
        + filteredImages.size());
}
```

Return an output stream consisting of the images that were downloaded from the URLs in the input stream

=

# of output stream elements must match the # of input stream elements

Implentating a Parallel Stream in ImageStreamGang

- We focus on processStream() in ImageStreamParallel.java

```
void processStream() {
  List<URL> urls = getInput();

  List<Image> filteredImages = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::blockingDownload)
    .flatMap(this::applyFilters)
    .collect(toList());

  System.out.println(TAG
        + "Image(s) filtered = "
        + filteredImages.size());
}
```

```
Image blockingDownload
              (URL url) {
  return BlockingTask
   .callInManagedBlock
     (() ->
       downloadImage(url));
}
```

See app/src/main/java/livelessons/imagestreamgang/streams/ImageStreamParallel.java

Implentating a Parallel Stream in ImageStreamGang

- We focus on processStream() in ImageStreamParallel.java

```
void processStream() {
   List<URL> urls = getInput();

   List<Image> filteredImages = urls
     .parallelStream()
     .filter(not(this::urlCached))
     .map(this::blockingDownload)
     .flatMap(this::applyFilters)
     .collect(toList());

   System.out.println(TAG
         + "Image(s) filtered = "
         + filteredImages.size());
}
```

```
Image blockingDownload
                (URL url) {
   return BlockingTask
    .callInManagedBlock
      (() ->
        downloadImage(url));
}
```

We cover what BlockingTask.callInManagedBlock() does in Part 3 of this lesson..

Implentating a Parallel Stream in ImageStreamGang

- We focus on processStream() in ImageStreamParallel.java

```
void processStream() {
   List<URL> urls = getInput();

   List<Image> filteredImages = urls
      .parallelStream()
      .filter(not(this::urlCached))
      .map(this::blockingDownload)
      .flatMap(this::applyFilters)
      .collect(toList());

   System.out.println(TAG
            + "Image(s) filtered = "
            + filteredImages.size());
}
```

Return an output stream containing the results of applying a list of filters to each image in the input stream & storing the results in the file system

See docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#flatMap

Implentating a Parallel Stream in ImageStreamGang

- We focus on processStream() in ImageStreamParallel.java

```
void processStream() {
  List<URL> urls = getInput();

  List<Image> filteredImages = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::blockingDownload)
    .flatMap(this::applyFilters)
    .collect(toList());

  System.out.println(TAG
         + "Image(s) filtered = "
         + filteredImages.size());
}
```

Return an output stream containing the results of applying a list of filters to each image in the input stream & storing the results in the file system

# of output stream elements may differ from the # of input stream elements

Implentating a Parallel Stream in ImageStreamGang

- We focus on processStream() in ImageStreamParallel.java

```
Stream<Image> applyFilters
                  (Image image) {
  return mFilters
    .parallelStream()

    .map(filter ->
         makeFilterWithImage
          (filter,
           image).run())
}
```

```
void processStream() {
  List<URL> urls = getInput();

  List<Image> filteredImages = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::blockingDownload)
    .flatMap(this::applyFilters)
    .collect(toList());

  System.out.println(TAG
        + "Image(s) filtered = "
        + filteredImages.size());
}
```

See app/src/main/java/livelessons/imagestreamgang/streams/ImageStreamParallel.java

Implentating a Parallel Stream in ImageStreamGang

- We focus on processStream() in ImageStreamParallel.java

```
void processStream() {
  List<URL> urls = getInput();

  List<Image> filteredImages = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::blockingDownload)
    .flatMap(this::applyFilters)
    .collect(toList());

  System.out.println(TAG
        + "Image(s) filtered = "
        + filteredImages.size());
}
```

Terminal operation triggers stream processing & yields a list result

See docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#collect

Implentating a Parallel Stream in ImageStreamGang

- We focus on processStream() in ImageStreamParallel.java

```
void processStream() {
  List<URL> urls = getInput();

  List<Image> filteredImages = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::blockingDownload)
    .flatMap(this::applyFilters)
    .collect(toList());

  System.out.println(TAG
         + "Image(s) filtered = "
         + filteredImages.size());
}
```

Terminal operation triggers stream processing & yields a list result

collect() is a "reduction" operation that combines elements into one result

Implentating a Parallel Stream in ImageStreamGang

- We focus on processStream() in ImageStreamParallel.java

```
void processStream() {
  List<URL> urls = getInput();

  List<Image> filteredImages = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::blockingDownload)
    .flatMap(this::applyFilters)
    .collect(toList());

  System.out.println(TAG
        + "Image(s) filtered = "
        + filteredImages.size());
}
```

Writes out the # of images downloaded, filtered, & stored

End of Java 8 Parallel ImageStreamGang Example (Part 2)