

The Java CompletableFuture ImageStreamGang Case Study: Applying Arbitrary-Arity Methods

Douglas C. Schmidt

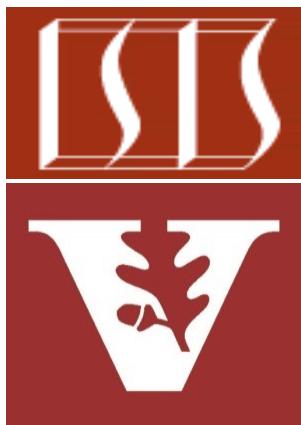
d.schmidt@vanderbilt.edu

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 design of the Java completable future version of ImageStreamGang
- Know how to apply completable futures to ImageStreamGang, e.g.
 - Factory methods
 - Completion stage methods
 - Arbitrary-arity methods
 - Via a wrapper for the allOf() method

```
<<Java Class>>
G CompletableFuture<T>

C CompletableFuture()
C cancel(boolean):boolean
C isCancelled():boolean
C isDone():boolean
C get()
C get(long,TimeUnit)
C join()
C complete(T):boolean
S supplyAsync(Supplier<U>):CompletableFuture<U>
S supplyAsync(Supplier<U>,Executor):CompletableFuture<U>
S runAsync(Runnable):CompletableFuture<Void>
S runAsync(Runnable,Executor):CompletableFuture<Void>
S completedFuture(U):CompletableFuture<U>
C thenApply(Function<?>):CompletableFuture<U>
C thenAccept(Consumer<? super T>):CompletableFuture<Void>
C thenCombine(CompletionStage<? extends U>,BiFunction<?>):CompletableFuture<V>
C thenCompose(Function<?>):CompletableFuture<U>
C whenComplete(BiConsumer<?>):CompletableFuture<T>
S allOf(CompletableFuture[]<?>):CompletableFuture<Void>
S anyOf(CompletableFuture[]<?>):CompletableFuture<Object>
```

Applying Arbitrary-Arity Methods in ImageStreamGang

Applying Arbitrary-Arity Methods in ImageStreamGang

- `collect()` returns a future to a stream of futures to images being processed asynchronously



flatMap() outputs a stream of futures associated with processing that's running asynchronously

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

```
    CompletableFuture<Stream<Image>>  
        resultsFuture = urls  
            .stream()  
            .map(this::checkUrlCachedAsync)  
            .map(this::downloadImageAsync)  
            .flatMap(this::applyFiltersAsync)  
            .collect(toFuture())  
            .thenApply(stream ->  
                log(stream.flatMap  
                    (Optional::stream),  
                    urls.size()))  
            .join();
```

Applying Arbitrary-Arity Methods in ImageStreamGang

- collect() returns a future to a stream of futures to images being processed asynchronously



Provides a single means to reactively await the completion of a stream of futures before continuing

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

```
    CompletableFuture<Stream<Image>>  
        resultsFuture = urls  
            .stream()  
            .map(this::checkUrlCachedAsync)  
            .map(this::downloadImageAsync)  
            .flatMap(this::applyFiltersAsync)  
            .collect(toFuture())  
            .thenApply(stream ->  
                log(stream.flatMap  
                    (Optional::stream),  
                    urls.size()))  
            .join();
```

Applying Arbitrary-Arity Methods in ImageStreamGang

- collect() returns a future to a stream of futures to images being processed asynchronously



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

```
    CompletableFuture<Stream<Image>>  
        resultsFuture = urls  
            .stream()  
            .map(this::checkUrlCachedAsync)  
            .map(this::downloadImageAsync)  
            .flatMap(this::applyFiltersAsync)  
            .collect(toFuture())  
            .thenApply(stream ->  
                log(stream.flatMap  
                    (Optional::stream),  
                    urls.size()))  
            .join();
```

collect() also triggers processing of all the intermediate operations

Applying Arbitrary-Arity Methods in ImageStreamGang

- collect() returns a future to a stream of futures to images being processed asynchronously
- StreamOfFuturesCollector wraps "arbitrary-arity" allOf() method

```
void processStream() {  
    List<URL> urls = getInput();  
  
    CompletableFuture<Stream<Image>>  
        resultsFuture = urls  
            .stream()  
            .map(this::checkUrlCachedAsync)  
            .map(this::downloadImageAsync)  
            .flatMap(this::applyFiltersAsync)  
            .collect(toFuture())  
            .thenApply(stream ->  
                log(stream.flatMap  
                    (Optional::stream),  
                    urls.size()))  
            .join();  
}
```

Return a future that completes when all futures in the stream complete

See [AndroidGUI/app/src/main/java/livelessons/utils/StreamOfFuturesCollector.java](#)

Applying Arbitrary-Arity Methods in ImageStreamGang

- collect() returns a future to a stream of futures to images being processed asynchronously
- StreamOfFuturesCollector wraps "arbitrary-arity" allOf() method

```
void processStream() {  
    List<URL> urls = getInput();  
  
    CompletableFuture<Stream<Image>>  
        resultsFuture = urls  
            .stream()  
            .map(this::checkUrlCachedAsync)  
            .map(this::downloadImageAsync)  
            .flatMap(this::applyFiltersAsync)  
            .collect(toFuture())  
            .thenApply(stream ->  
                log(stream.flatMap  
                    (Optional::stream),  
                    urls.size()))  
            .join();  
}
```

*Log the results after the
final future completes*

Applying Arbitrary-Arity Methods in ImageStreamGang

- collect() returns a future to a stream of futures to images being processed asynchronously
- StreamOfFuturesCollector wraps "arbitrary-arity" allOf() method

```
void processStream() {  
    List<URL> urls = getInput();  
  
    CompletableFuture<Stream<Image>>  
        resultsFuture = urls  
            .stream()  
            .map(this::checkUrlCachedAsync)  
            .map(this::downloadImageAsync)  
            .flatMap(this::applyFiltersAsync)  
            .collect(toFuture())  
            .thenApply(stream ->  
                log(stream.flatMap  
                    (Optional::stream),  
                    urls.size()))  
            .join();  
}
```

*Remove empty optional values
from the stream in Java 9+*

See docs.oracle.com/javase/9/docs/api/java/util/Optional.html#flatMap

Applying Arbitrary-Arity Methods in ImageStreamGang

- collect() returns a future to a stream of futures to images being processed asynchronously
- StreamOfFuturesCollector wraps "arbitrary-arity" allOf() method

```
void processStream() {  
    List<URL> urls = getInput();  
  
    CompletableFuture<Stream<Image>>  
        resultsFuture = urls  
            .stream()  
            .map(this::checkUrlCachedAsync)  
            .map(this::downloadImageAsync)  
            .flatMap(this::applyFiltersAsync)  
            .collect(toFuture())  
            .thenApply(stream -> log(stream  
                .filter(Optional::isPresent)  
                .map(Optional::get),  
                    urls.size()))  
            .join();  
}
```

*Remove empty optional values
from the stream in Java 8*

Applying Arbitrary-Arity Methods in ImageStreamGang

- collect() returns a future to a stream of futures to images being processed asynchronously
- StreamOfFuturesCollector wraps "arbitrary-arity" allOf() method



VERBOSE

blah blah blah
blah blah blah
blah blah blah
blah blah blah
blah blah bla
blah blah blah

Java 8 is more verbose..

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

```
    CompletableFuture<Stream<Image>>  
        resultsFuture = urls  
            .stream()  
            .map(this::checkUrlCachedAsync)  
            .map(this::downloadImageAsync)  
            .flatMap(this::applyFiltersAsync)  
            .collect(toFuture())  
            .thenApply(stream -> log(stream  
                .filter(Optional::isPresent)  
                .map(Optional::get),  
                    urls.size()))  
            .join();
```

See blog.codefx.org/java/java-9-optional

Applying Arbitrary-Arity Methods in ImageStreamGang

- collect() returns a future to a stream of futures to images being processed asynchronously
- StreamOfFuturesCollector wraps "arbitrary-arity" allOf() method

```
void processStream() {  
    List<URL> urls = getInput();  
  
    CompletableFuture<Stream<Image>>  
        resultsFuture = urls  
            .stream()  
            .map(this::checkUrlCachedAsync)  
            .map(this::downloadImageAsync)  
            .flatMap(this::applyFiltersAsync)  
            .collect(toFuture())  
            .thenApply(stream ->  
                log(stream.flatMap  
                    (Optional::stream),  
                    urls.size()))  
            .join();  
}
```

*Wait until all the async
processing is completed*



Applying Arbitrary-Arity Methods in ImageStreamGang

- collect() returns a future to a stream of futures to images being processed asynchronously



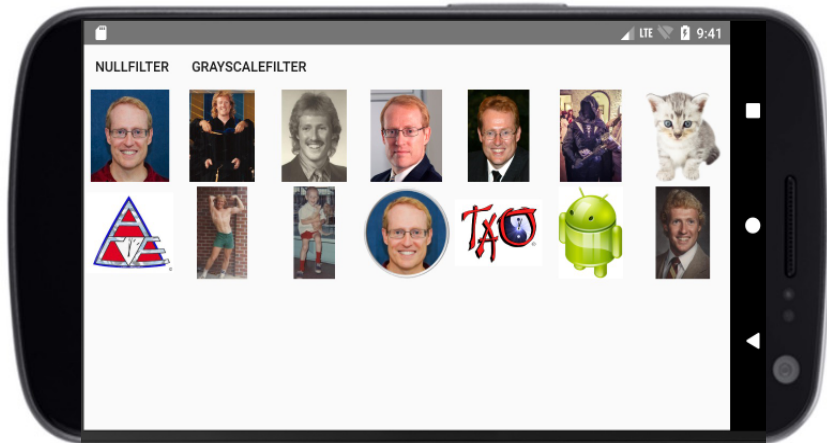
```
void processStream() {  
    List<URL> urls = getInput();
```

```
    CompletableFuture<Stream<Image>>  
        resultsFuture = urls  
            .stream()  
            .map(this::checkUrlCachedAsync)  
            .map(this::downloadImageAsync)  
            .flatMap(this::applyFiltersAsync)  
            .collect(toFuture())  
            .thenApply(stream ->  
                log(stream.flatMap  
                    (Optional::stream),  
                    urls.size()))  
            .join();
```

This is the one & only call to join() in this async stream pipeline!

Applying Arbitrary-Arity Methods in ImageStreamGang

- collect() returns a future to a stream of futures to images being processed asynchronously



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

```
    CompletableFuture<Stream<Image>>  
        resultsFuture = urls  
            .stream()  
            .map(this::checkUrlCachedAsync)  
            .map(this::downloadImageAsync)  
            .flatMap(this::applyFiltersAsync)  
            .collect(toFuture())  
            .thenApply(stream ->  
                log(stream.flatMap  
                    (Optional::stream),  
                    urls.size()))  
            .join();
```

Images are displayed after all the async processing completes

End of the Java Completable Future ImageStreamGang Case Study: Applying Arbitrary-Arity Methods