Advanced Java Completable Future Features: Implementing the FuturesCollector Class

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 how arbitrary-arity methods process Completable Future objects in bulk
- Recognize the limitations with these methods
- Know how to address these limitations by wrapping the allOf() method to work seamlessly with Java's Streams framework

```
class FuturesCollector<T>
      implements Collector
  <CompletableFuture<T>,
   List<CompletableFuture<T>>,
   CompletableFuture<List<T>>> {
  ...
```

- Learn how to implement the FuturesCollector class
  - Integrates the arbitrary-arity method allOf() into the Java Streams collector framework & returns a CompletableFuture to a List of objects that are being processed asynchronously & concurrently

See Java8/ex8/src/main/java/utils/FuturesCollector.java

Implementing the FuturesCollector Class

Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

```
public class FuturesCollector<T>
       implements Collector<CompletableFuture<T>,
                            List<CompletableFuture<T>>,
                            CompletableFuture<List<T>>> {
   ...
```

Implements a custom Collector

See docs.oracle.com/javase/8/docs/api/java/util/stream/Collector.html

Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

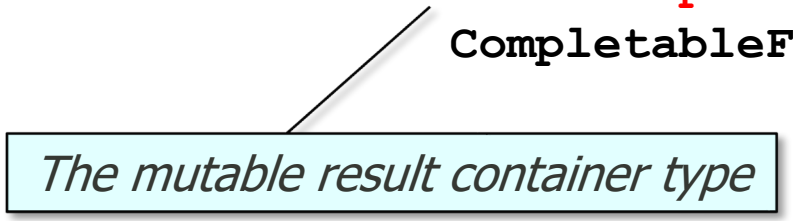
```
public class FuturesCollector<T>
       implements Collector<CompletableFuture<T>,
                            List<CompletableFuture<T>>,
                            CompletableFuture<List<T>>> {
   ...
```

The type of input elements in the stream

Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

```
public class FuturesCollector<T>
       implements Collector<CompletableFuture<T>,
                            List<CompletableFuture<T>>,
                            CompletableFuture<List<T>>> {
   ...
```



Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

```
public class FuturesCollector<T>
       implements Collector<CompletableFuture<T>,
                            List<CompletableFuture<T>>,
                            CompletableFuture<List<T>>> {
   ...
```

The result type of final output of the Collector

Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

```
public class FuturesCollector<T>
       implements Collector<CompletableFuture<T>,
                            List<CompletableFuture<T>>,
                            CompletableFuture<List<T>>> {
  public Supplier<List<CompletableFuture<T>>> supplier() {
    return ArrayList::new;
  }

  public BiConsumer<List<CompletableFuture<T>>,
                    CompletableFuture<T>> accumulator()
  { return List::add; }
  ...
```

This factory method returns a Supplier used by the Java Streams Collector framework to create a new mutable ArrayList container

Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

```
public class FuturesCollector<T>
       implements Collector<CompletableFuture<T>,
                            List<CompletableFuture<T>>,
                            CompletableFuture<List<T>>> {
  public Supplier<List<CompletableFuture<T>>> supplier() {
    return ArrayList::new;
  }

  public BiConsumer<List<CompletableFuture<T>>,
                    CompletableFuture<T>> accumulator()
  { return List::add; }
  ...
```

This mutable result container stores a List of CompletableFuture objects of type T

Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

```
public class FuturesCollector<T>
       implements Collector<CompletableFuture<T>,
                            List<CompletableFuture<T>>,
                            CompletableFuture<List<T>>> {
  public Supplier<List<CompletableFuture<T>>> supplier() {
    return ArrayList::new;
  }
```

This factory method returns a BiConsumer used by the Java Streams Collector framework to add a new CompletableFuture into the mutable ArrayList container

```
  public BiConsumer<List<CompletableFuture<T>>,
                    CompletableFuture<T>> accumulator()
  { return List::add; }
  ...
```

This method is only ever called in a single thread (so no locks are needed)

Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

```
public class FuturesCollector<T>
   ...
   public BinaryOperator<List<CompletableFuture<T>>> combiner() {
     return (List<CompletableFuture<T>> one,
             List<CompletableFuture<T>> another) -> {
         one.addAll(another);
         return one;
     };
   }
   ...
```

This factory method returns a BinaryOperator that merges two partial ArrayList results into a single ArrayList (only relevant for parallel streams)

This method is only ever called in a single thread (so no locks are needed)

Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

```
public class FuturesCollector<T>
  ...
  public Function<List<CompletableFuture<T>>,
                  CompletableFuture<List<T>>> finisher() {
    return futures -> CompletableFuture
      .allOf(futures.toArray(new CompletableFuture[0]))
      .thenApply(v -> futures.stream()
                             .map(CompletableFuture::join)
                             .toList());
  }
  ...
```

This factory method returns a Function used by the Java Streams Collector framework to transform the ArrayList mutable result container to the completable future result type

Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

```
public class FuturesCollector<T>
  ...
  public Function<List<CompletableFuture<T>>,
                  CompletableFuture<List<T>>> finisher() {
    return futures -> CompletableFuture
      .allOf(futures.toArray(new CompletableFuture[0]))

      .thenApply(v -> futures.stream()
                             .map(CompletableFuture::join)
                             .toList());
  }
  ...
```

Reference to the mutable result container, which is an ArrayList.

Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

```
public class FuturesCollector<T>
  ...
  public Function<List<CompletableFuture<T>>,
                  CompletableFuture<List<T>>> finisher() {
    return futures -> CompletableFuture
      .allOf(futures.toArray(new CompletableFuture[0]))
      .thenApply(v -> futures.stream()
                             .map(CompletableFuture::join)
                             .toList());
  }
  ...
```

Convert the List of futures to an array of futures & pass to allOf() to obtain a future that will complete when all futures complete.

Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

```
public class FuturesCollector<T>
   ...
   public Function<List<CompletableFuture<T>>,
                           CompletableFuture<List<T>>> finisher() {
     return futures -> CompletableFuture
       .allOf(futures.toArray(new CompletableFuture[0]))
       .thenApply(v -> futures.stream()
                              .map(CompletableFuture::join)
                              .toList());
   }
   ...
```

When all futures have completed get a single future to a List of joined elements of type T.

Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

```
public class FuturesCollector<T>
   ...
   public Function<List<CompletableFuture<T>>,
                           CompletableFuture<List<T>>> finisher() {
     return futures -> CompletableFuture
       .allOf(futures.toArray(new CompletableFuture[0]))

       .thenApply(v -> futures.stream()
                                .map(CompletableFuture::join)
                                .toList());
   }
   ...
```

Convert the ArrayList of futures into a Stream of futures

Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

```
public class FuturesCollector<T>
  ...
  public Function<List<CompletableFuture<T>>,
                  CompletableFuture<List<T>>> finisher() {
    return futures -> CompletableFuture
      .allOf(futures.toArray(new CompletableFuture[0]))

      .thenApply(v -> futures.stream()
                             .map(CompletableFuture::join)
                             .toList());
  }
  ...
```

This call to join() will never block!

Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

```
public class FuturesCollector<T>
   ...
   public Function<List<CompletableFuture<T>>,
                                CompletableFuture<List<T>>> finisher() {
      return futures -> CompletableFuture
         .allOf(futures.toArray(new CompletableFuture[0]))

         .thenApply(v -> futures.stream()
                                .map(CompletableFuture::join)
                                .toList());
   }
   ...
```

Return a future to a list of elements of T

Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

```
public class FuturesCollector<T>
  ...
  public Set<Characteristics> characteristics() {
    return Collections.singleton(Characteristics.UNORDERED);
  }

  public static <T> Collector<CompletableFuture<T>, ?,
                              CompletableFuture<List<T>>>
  toFuture() {
    return new FuturesCollector<>();
  }
}
```

Returns a set indicating the FuturesCollector characteristics

FuturesCollector is thus a *non-concurrent* collector

Implementing the FuturesCollector Class

- FuturesCollector implements all methods in the Collector interface

```
public class FuturesCollector<T>
  ...
  public Set<Characteristics> characteristics() {
    return Collections.singleton(Characteristics.UNORDERED);
  }

  public static <T> Collector<CompletableFuture<T>, ?,
                              CompletableFuture<List<T>>>
  toFuture() {
    return new FuturesCollector<>();
  }
}
```

This static factory method creates a new FuturesCollector

Implementing the FuturesCollector Class

- FuturesCollector is used to return a completable future to a list of big fractions that are being reduced & multiplied asynchronously

```
static void testFractionMultiplications1() {
   ...
   Stream.generate(() -> makeBigFraction(new Random(), false))

         .limit(sMAX_FRACTIONS)

         .map(reduceAndMultiplyFraction)

         .collect(FuturesCollector.toFuture())

         .thenAccept(this::sortAndPrintList);
}
```

toFuture() returns an instance of the FuturesCollector for use with collect()

See lesson on "*Advanced Java Completable Future Features: Applying Completion Stage Methods*"

Implementing the FuturesCollector Class

- FuturesCollector is used to return a completable future to a list of big fractions that are being reduced & multiplied asynchronously

```
static void testFractionMultiplications1() {
   ...
   Stream.generate(() -> makeBigFraction(new Random(), false))

         .limit(sMAX_FRACTIONS)

         .map(reduceAndMultiplyFraction)

         .collect(FuturesCollector.toFuture())

         .thenAccept(this::sortAndPrintList);
}
```

thenAccept() is called only after the future returned from collect() completes

See lesson on "*Advanced Java Completable Future Features: Applying Completion Stage Methods*"

End of Advanced Java Completable Future Features: Implementing the FuturesCollector Class