Learning Objectives in this Part of the Lesson

• Understand advanced features of completable futures, e.g.
  • Factory methods that initiate async functionality
  • Completion stage methods used to chain together actions that perform async result processing & composition

• Arbitrary-arity methods that process futures in bulk

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html
Arbitrary-Arity Methods
Process Futures in Bulk
Arbitrary-Arity Methods Process Futures in Bulk

- Arbitrary-arity methods are triggered after completion of some/all of \( n \) futures

<table>
<thead>
<tr>
<th>Methods</th>
<th>Params</th>
<th>Returns</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>allOf</td>
<td>Varargs</td>
<td>Completable Future&lt;Void&gt;</td>
<td>Return a future that completes when all futures in params complete</td>
</tr>
<tr>
<td>anyOf</td>
<td>Varargs</td>
<td>Completable Future&lt;Void&gt;</td>
<td>Return a future that completes when any future in params complete</td>
</tr>
</tbody>
</table>

These “arbitrary-arity” methods are hard to program without using wrappers.
• Arbitrary-arity methods are triggered after completion of some/all of \( n \) futures

See [en.wikipedia.org/wiki/Arity](http://en.wikipedia.org/wiki/Arity)
Arbitrary-arity methods are triggered after completion of some/all of $n$ futures

- Can wait for any or all completable futures in an array to complete
Arbitrary-Arity Methods Process Futures in Bulk

- Arbitrary-arity methods are triggered after completion of some/all of \( n \) futures
- Can wait for any or all completable futures in an array to complete

We focus on `allOf()`

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#allOf](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#allOf)
FuturesCollector is used to return a completable future to a list of big fractions that are being reduced and multiplied asynchronously.

```java
static void testFractionMultiplications() {
    ... 
    Stream.generate(() -> makeBigFraction(new Random(), false))
        .limit(sMAX_FRACTIONS)
        .map(reduceAndMultiplyFraction)
        .collect(FuturesCollector.toFuture())
        .thenAccept(printSortedList);
}
```

See [github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex8](https://github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex8)
Arbitrary-Arity Methods Process Futures in Bulk

- FuturesCollector provides a wrapper for allOf()

```java
<<Java Interface>>
Collector<T,A,R>
- supplier(): Supplier<A>
- accumulator(): BiConsumer<A,T>
- combiner(): BinaryOperator<A>
- finisher(): Function<A,R>
- characteristics(): Set<Characteristics>

<<Java Class>>
FuturesCollector<T>
- FuturesCollector()
- supplier()
- accumulator()
- combiner()
- finisher()
- characteristics()
- toFuture(): Collector<CompletableFuture<T>, ?, CompletableFuture<List<T>>>

See Java8/ex8/utils/FuturesCollector.java
```
Arbitrary-Arity Methods Process Futures in Bulk

- FuturesCollector provides a wrapper for `allOf()`
- Converts a *stream* of completable futures into a *single* completable future that’s triggered when *all* futures in the stream complete

```java
<<Java Interface>>
Collector<T,A,R>
- supplier(): Supplier<A>
- accumulator(): BiConsumer<A,T>
- combiner(): BinaryOperator<A>
- finisher(): Function<A,R>
- characteristics(): Set<Characteristics>

<<Java Class>>
FuturesCollector<T>
- FuturesCollector()
- supplier()
- accumulator()
- combiner()
- finisher()
- characteristics()
- toFuture(): Collector<CompletableFuture<T>, ?, CompletableFuture<List<T>>>
Arbitrary-Arity Methods Process Futures in Bulk

- FuturesCollector provides a wrapper for allOf()
  - Converts a stream of completable futures into a single completable future that’s triggered when all futures in the stream complete
- Implements the Collector interface that accumulates input elements into a mutable result container

See docs.oracle.com/javase/8/docs/api/java/util/stream/Collector.html
Arbitrary-Arity Methods Process Futures in Bulk

• FuturesCollector provides a wrapper for allOf()

**Java Interface**

```
Collector<T,A,R>
```
- supplier(): Supplier<A>
- accumulator(): BiConsumer<A,T>
- combiner(): BinaryOperator<A>
- finisher(): Function<A,R>
- characteristics(): Set<Characteristics>

**Java Class**

```
FuturesCollector<T>
```
- FuturesCollector()
- supplier()
- accumulator()
- combiner()
- finisher()
- characteristics()
- toFuture(): Collector<CompletableFuture<T>, ?, CompletableFuture<List<T>>>

FuturesCollector provides a powerful wrapper for some complex code!!!
Arbitrary-Arity Methods Process Futures in Bulk

- FuturesCollector provides a wrapper for allOf()

```java
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](docs.oracle.com/javase/8/docs/api/java/util/stream/Collector.html)
Arbitrary-Arity Methods Process Futures in Bulk

• FuturesCollector provides a wrapper for allOf()

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

The type of input elements to the accumulator() method
Arbitrary-Arity Methods Process Futures in Bulk

- FuturesCollector provides a wrapper for allOf()

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

The mutable accumulation type of the accumulator() method
FuturesCollector provides a wrapper for allOf()

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

The result type of the finisher() method, i.e., the final output of the collector
Arbitrary-Arity Methods Process Futures in Bulk

- FuturesCollector provides a wrapper for allOf()

```java
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;
    }

    ...}
```

*Factory method that creates & returns a new mutable array list container*
FuturesCollector provides a wrapper for allOf()

```java
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;
    }
    ...
```

Folds a new completable future into the mutable array list container
Arbitrary-Arity Methods Process Futures in Bulk

- FuturesCollector provides a wrapper for allOf()

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

Accepts two partial array list results & merges them into a single array list
FuturesCollector provides a wrapper for allOf()

```java
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)
                .collect(toList()));
    }
    ...
}
```

Perform final transformation from the intermediate array list accumulation type to the final completable future result type.
• FuturesCollector provides a wrapper for allOf()

```java
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)
                .collect(toList()));
    }
```

Convert list of futures to array of futures & pass to allOf() to obtain a future that will complete when all futures complete
Arbitrary-Arity Methods Process Futures in Bulk

• FuturesCollector provides a wrapper for allOf()

```java
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)
                .collect(toList()));
    }
    ...
}
```

When all futures have completed get a single future to a list of joined elements of type T
FuturesCollector provides a wrapper for allOf()

```java
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)
                .collect(toList()));
    }
    ...
}
```

*This call to join() will never block!*
Arbitrary-Arity Methods Process Futures in Bulk

- FuturesCollector provides a wrapper for allOf()

```java
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)
                     .collect(toList()));
}
```

Return a future to a list of elements of T

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

...
Arbitrary-Arity Methods Process Futures in Bulk

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

```java
static void testFractionMultiplications() {
    ...
    Stream.generate(() -> makeBigFraction(new Random(), false))
        .limit(sMAX_FRACTIONS)
        .map(reduceAndMultiplyFraction)
        .collect(FuturesCollector.toFuture())
        .thenAccept(printSortedList);
}
```

See [github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex8](https://github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex8)
FuturesCollector provides a wrapper for allOf()

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

public static <T> Collector<CompletableFuture<T>, ?, CompletableFuture<List<T>>>
    toFuture() {
    return new FuturesCollector<>();
}
```
Arbitrary-Arity Methods Process Futures in Bulk

- FuturesCollector provides a wrapper for allOf()

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

Static factory method creates a new FuturesCollector

```java
public static <T> Collector<CompletableFuture<T>, ?, CompletableFuture<List<T>>>
toFuture() {
    return new FuturesCollector<>();
}
```
End of Overview of Advanced Java 8 Completable Future Features (Part 4)