

Advanced Java Completable Future

Features: Arbitrary-Arity Methods

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

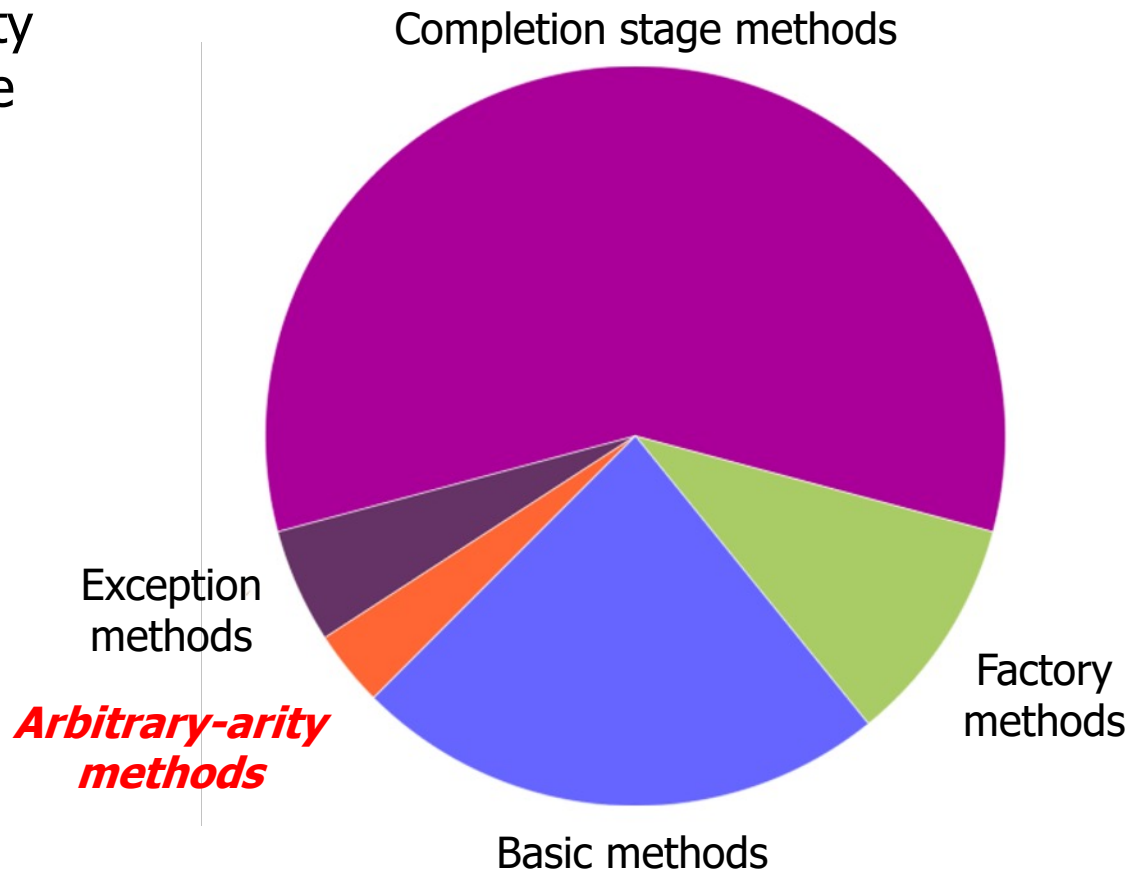
**Institute for Software
Integrated Systems**

**Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Part of the Lesson

- Understand how arbitrary-arity methods process Completable Future objects in bulk



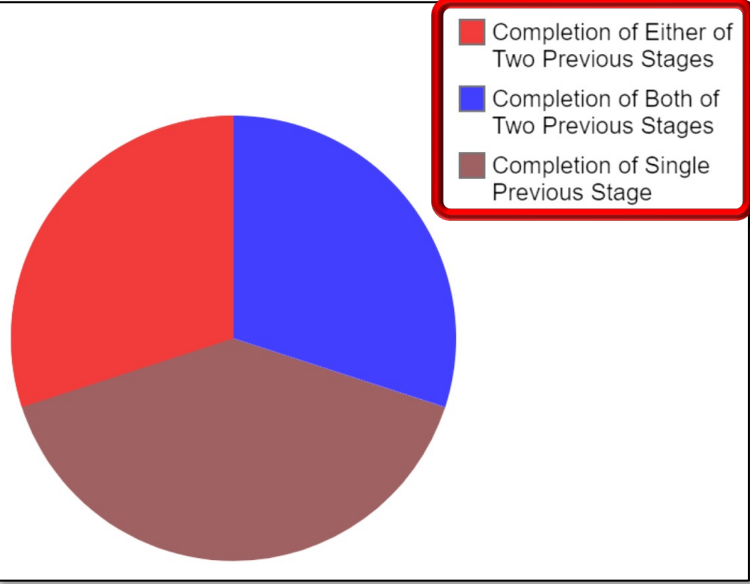
See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html

Arbitrary-Arity Methods Process Futures in Bulk

Arbitrary-Arity Methods Process Futures in Bulk

- Completion stage methods can only be triggered by one and/or two previous stage(s)

Completion stage methods

- 
- Completion of Either of Two Previous Stages
 - Completion of Both of Two Previous Stages
 - Completion of Single Previous Stage

Exception methods

Arbitrary-arity methods

Basic methods

Factory methods

See "Advanced Java Completable Future Features: Grouping Completion Stage Methods"

Arbitrary-Arity Methods Process Futures in Bulk

- Completion stage methods can only be triggered by one and/or two previous stage(s)
 - However, it's often necessary to trigger completion stage methods after an arbitrary number of futures complete

Stream

```
.generate(() -> makeBigFraction  
           (new Random(), false))  
  
.limit(sMAX_FRACTIONS)  
  
.map(reduceAndMultiplyFractions)  
  
.collect  
  (FuturesCollector.toFuture())  
  
.thenAccept  
  (this::sortAndPrintList);
```

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

Arbitrary-Arity Methods Process Futures in Bulk

- Completion stage methods can only be triggered by one and/or two previous stage(s)
- However, it's often necessary to trigger completion stage methods after an arbitrary number of futures complete

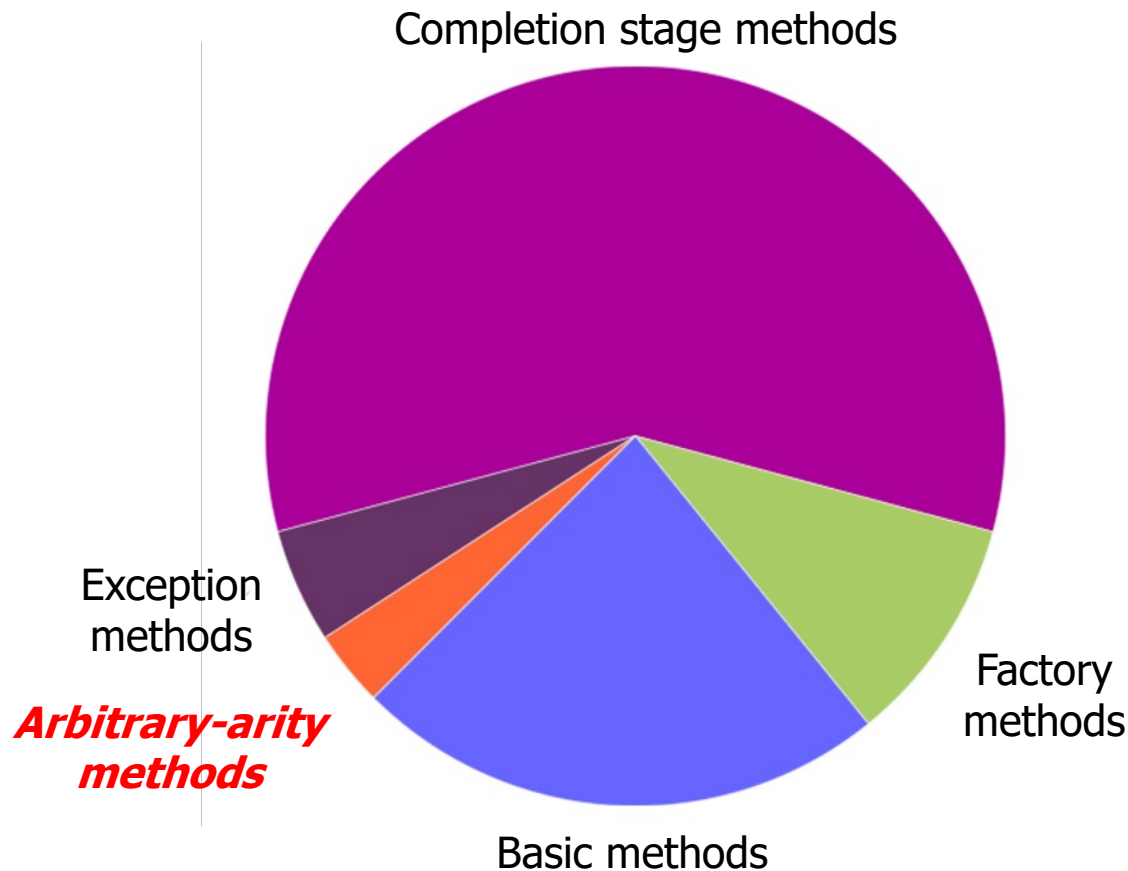
Need to convert a stream of `CompletableFuture` objects into a single `CompletableFuture`

Stream

```
.generate(() -> makeBigFraction  
            (new Random(), false))  
  
.limit(sMAX_FRACTIONS)  
  
.map(reduceAndMultiplyFractions)  
  
.collect  
  (FuturesCollector.toFuture())  
  
.thenAccept  
  (this::sortAndPrintList);
```

Arbitrary-Arity Methods Process Futures in Bulk

- An arbitrary-arity method returns one future that is triggered after completion of any/all futures



See en.wikipedia.org/wiki/Arity

Arbitrary-Arity Methods Process Futures in Bulk

- An arbitrary-arity method returns one future that is triggered after completion of any/all futures

Methods	Params	Returns	Behavior
<code>allOf</code>	<code>Varargs</code>	<code>Completable Future<Void></code>	Return a future that completes when all futures in params complete
<code>anyOf</code>	<code>Varargs</code>	<code>Completable Future<Void></code>	Return a future that completes when any future in params complete

See en.wikipedia.org/wiki/Arity

Arbitrary-Arity Methods Process Futures in Bulk

- An arbitrary-arity method returns one future that is triggered after completion of any/all futures

Methods	Params	Returns	Behavior
<code>allOf</code>	<code>Varargs</code>	<code>Completable Future<Void></code>	Return a future that completes when all futures in params complete
<code>anyOf</code>	<code>Varargs</code>	<code>Completable Future<Void></code>	Return a future that completes when any future in params complete

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#allOf

Arbitrary-Arity Methods Process Futures in Bulk

- An arbitrary-arity method returns one future that is triggered after completion of any/all futures

Methods	Params	Returns	Behavior
<code>allOf</code>	<code>Varargs</code>	<code>Completable Future<Void></code>	Return a future that completes when all futures in params complete
<code>anyOf</code>	<code>Varargs</code>	<code>Completable Future<Void></code>	Return a future that completes when any future in params complete

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#anyOf

Arbitrary-Arity Methods Process Futures in Bulk

- An arbitrary-arity method returns one future that is triggered after completion of any/all futures
- The returned future can be used to “wait” for any or all of the N completable futures in an array to complete

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

Arbitrary-Arity Methods Process Futures in Bulk

- An arbitrary-arity method returns one future that is triggered after completion of any/all futures
- The returned future can be used to “wait” for any or all of the N completable futures in an array to complete
 - This “wait” rarely blocks

<<Java Class>>

CompletableFuture<T>

CompletableFuture()

cancel(boolean):boolean

isCancelled():boolean

isDone():boolean

get()

get(long,TimeUnit)

join()

complete(T):boolean

^SsupplyAsync(Supplier<U>):CompletableFuture<U>

^SsupplyAsync(Supplier<U>,Executor):CompletableFuture<U>

^SrunAsync(Runnable):CompletableFuture<Void>

^SrunAsync(Runnable,Executor):CompletableFuture<Void>

^ScompletedFuture(U):CompletableFuture<U>

thenApply(Function<?>):CompletableFuture<U>

thenAccept(Consumer<? super T>):CompletableFuture<Void>

thenCombine(CompletionStage<? extends U>,BiFunction<?>):CompletableFuture<V>

thenCompose(Function<?>):CompletableFuture<U>

whenComplete(BiConsumer<?>):CompletableFuture<T>

^SallOf(CompletableFuture[]<?>):CompletableFuture<Void>

^SanyOf(CompletableFuture[]<?>):CompletableFuture<Object>

Arbitrary-Arity Methods Process Futures in Bulk

- An arbitrary-arity method returns one future that is triggered after completion of any/all futures
- The returned future can be used to “wait” for any or all of the N completable futures in an array to complete
 - This “wait” rarely blocks
 - Instead, completion stage methods process the results reactively

<<Java Class>>

CompletableFuture<T>

CompletableFuture()

cancel(boolean):boolean

isCancelled():boolean

isDone():boolean

get()

get(long,TimeUnit)

join()

complete(T):boolean

^SsupplyAsync(Supplier<U>):CompletableFuture<U>

^SsupplyAsync(Supplier<U>,Executor):CompletableFuture<U>

^SrunAsync(Runnable):CompletableFuture<Void>

^SrunAsync(Runnable,Executor):CompletableFuture<Void>

^ScompletedFuture(U):CompletableFuture<U>

thenApply(Function<?>):CompletableFuture<U>

thenAccept(Consumer<? super T>):CompletableFuture<Void>


thenCombine(CompletionStage<? extends U>,BiFunction<?>):CompletableFuture<V>

thenCompose(Function<?>):CompletableFuture<U>

whenComplete(BiConsumer<?>):CompletableFuture<T>

^SallOf(CompletableFuture[]<?>):CompletableFuture<Void>

^SanyOf(CompletableFuture[]<?>):CompletableFuture<Object>



Help make programs more *responsive* by not blocking caller code

Arbitrary-Arity Methods Process Futures in Bulk

- An arbitrary-arity method returns one future that is triggered after completion of any/all futures
- The returned future can be used to “wait” for any or all of the N completable futures in an array to complete
- We focus on `allOf()`, which is like `thenCombine()` on steroids!



<<Java Class>>	
G CompletableFuture<T>	
C	CompletableFuture()
•	cancel(boolean):boolean
•	isCancelled():boolean
•	isDone():boolean
•	get()
•	get(long,TimeUnit)
•	join()
•	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>
•	thenApply(Function<?>):CompletableFuture<U>
•	thenAccept(Consumer<? super T>):CompletableFuture<Void>
•	thenCombine(CompletionStage<? extends U>,BiFunction<?>):CompletableFuture<V>
•	thenCompose(Function<?>):CompletableFuture<U>
•	whenComplete(BiConsumer<?>):CompletableFuture<T>
S	allOf(CompletableFuture[]<?>):CompletableFuture<Void>
S	anyOf(CompletableFuture[]<?>):CompletableFuture<Object>

End of Advanced Java

Completable Future Features:

Arbitrary-Arity Methods