

Overview of Java 8 CompletableFutures (Part 2)

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

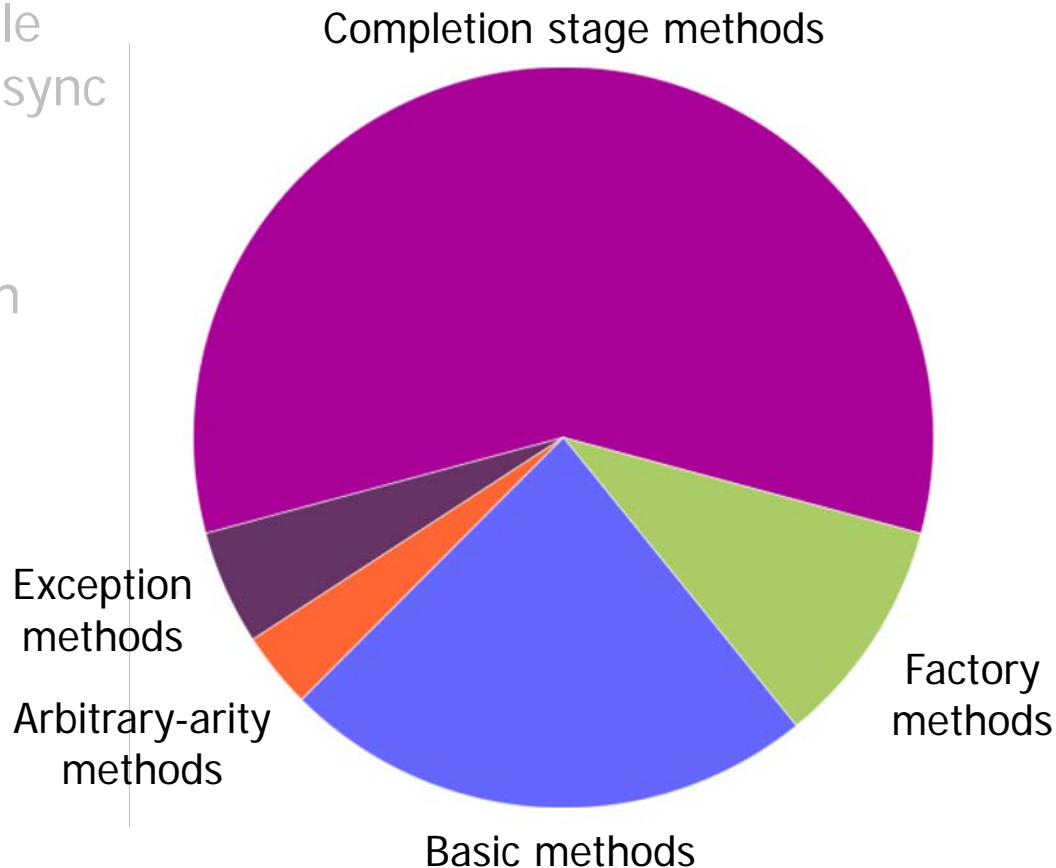
Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA



Learning Objectives in this Part of the Lesson

- Know how the Java 8 completable futures framework provides an async concurrent programming model
- Recognize Java 8 completable futures overcome limitations with Java futures
- Be able to group methods in the Java 8 completable future API



Grouping the Java 8 Completable Future API

Grouping the Java 8 Completable Future API

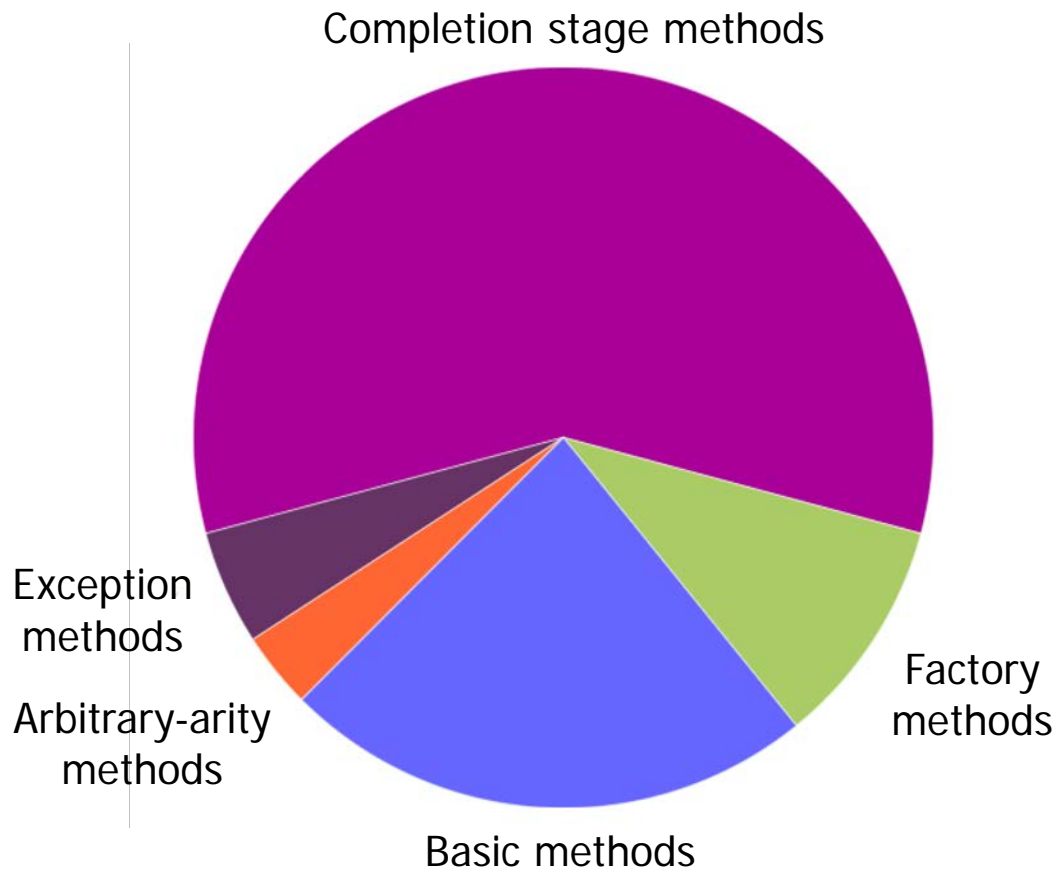
- The entire completable future framework resides in one class with 60+ methods!!!

<<Java Class>>	
G CompletableFuture<T>	
•	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>

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

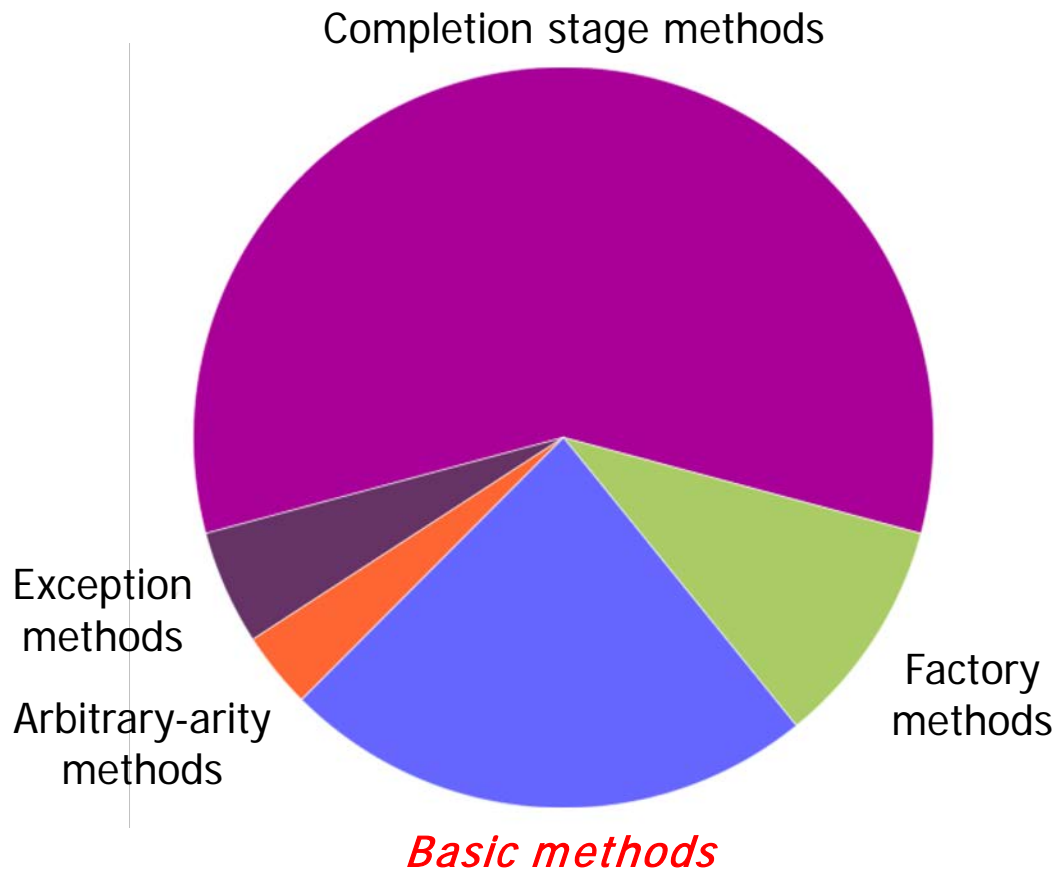
Grouping the Java 8 Completable Future API

- The entire completable future framework resides in one class with 60+ methods!!!
- It therefore helps to have a “birds-eye” view of this class



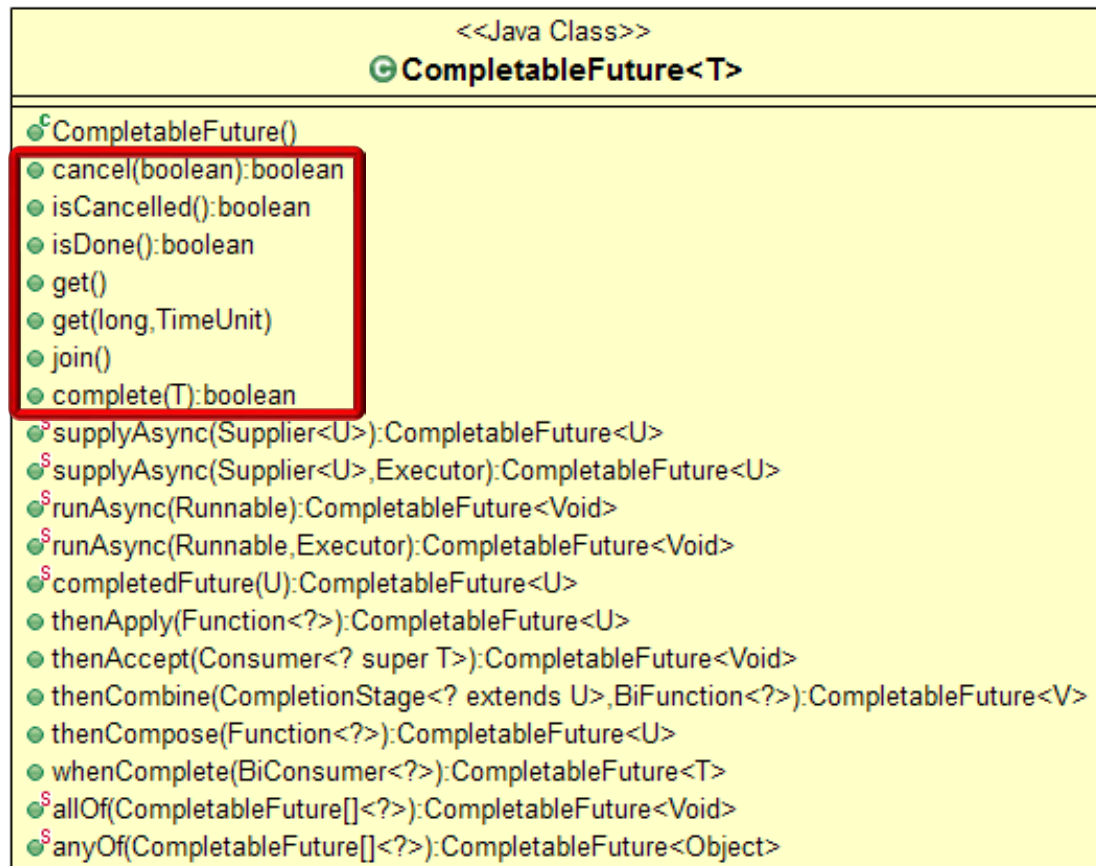
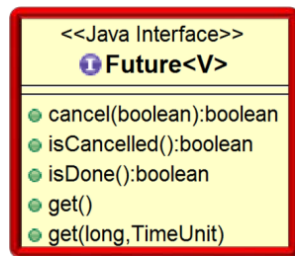
Grouping the Java 8 Completable Future API

- Some completable future features are basic



Grouping the Java 8 Completable Future API

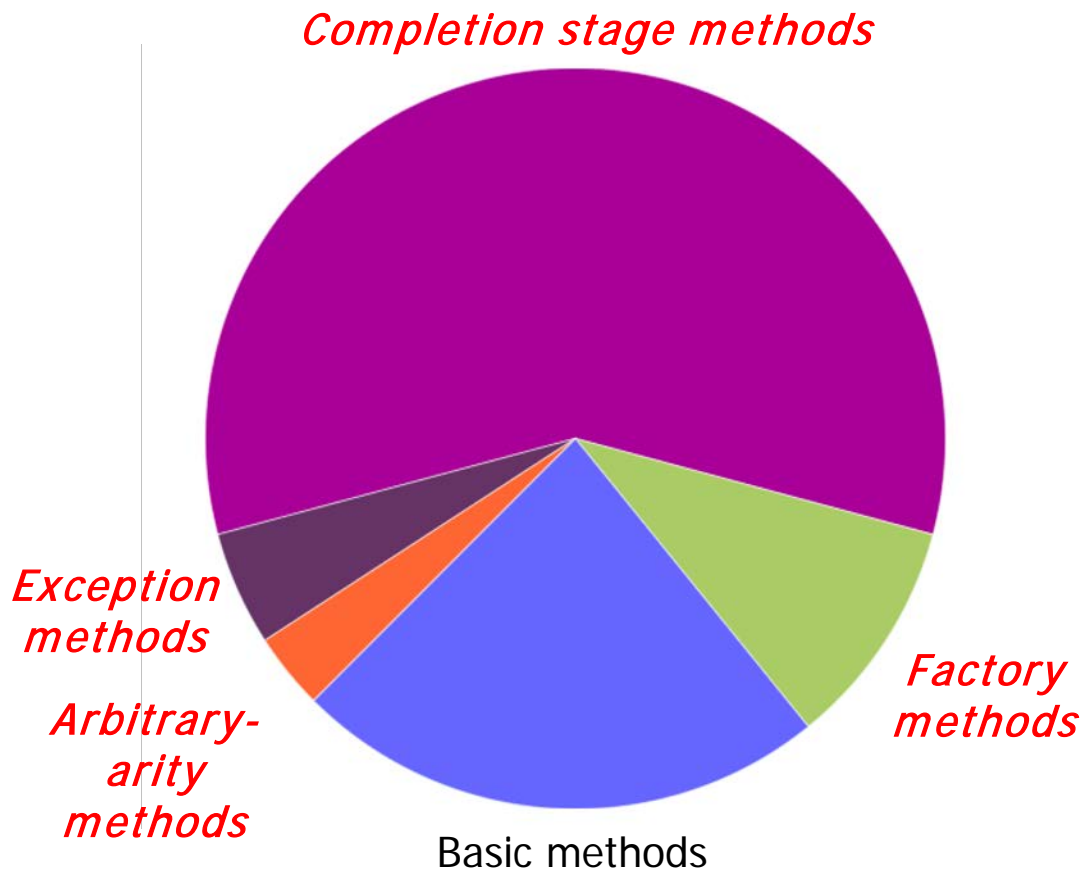
- Some completable future features are basic
- e.g., the Java Future API + some simple enhancements



Only slightly better than the conventional Future interface

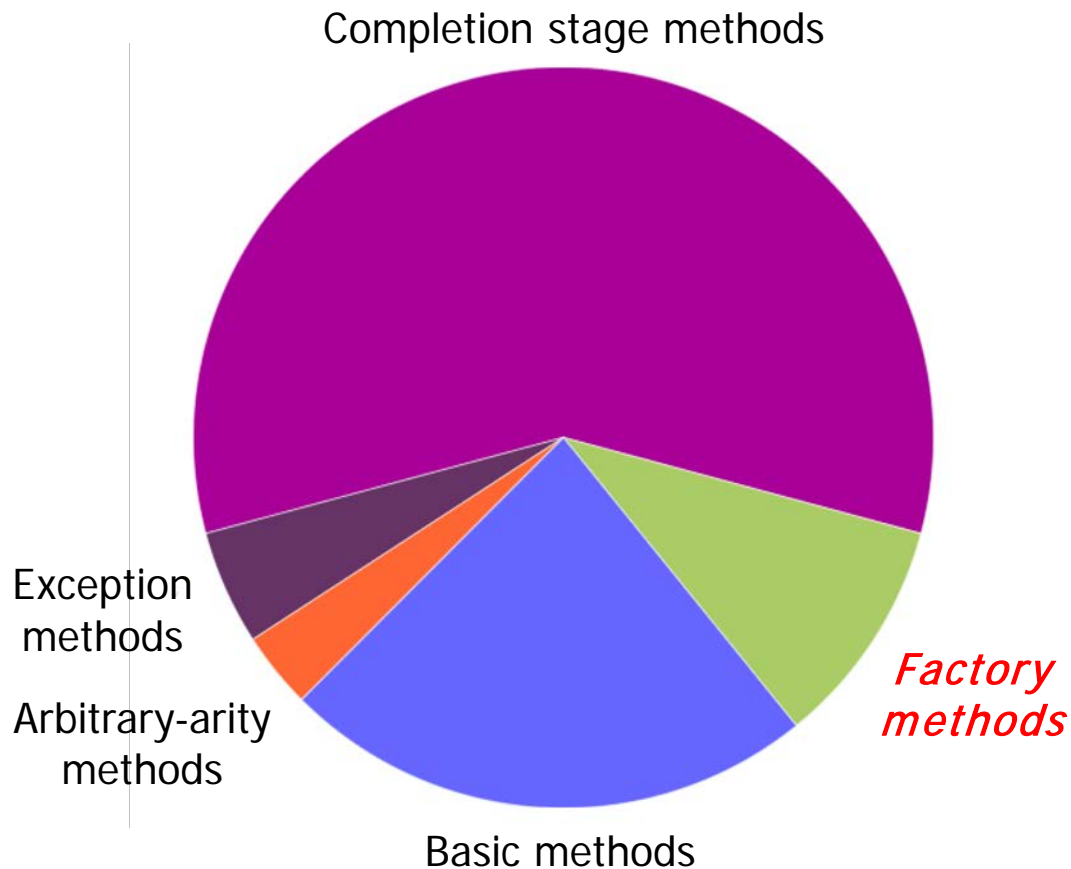
Grouping the Java 8 Completable Future API

- Other completable future features are more advanced



Grouping the Java 8 Completable Future API

- Other completable future features are more advanced
 - Factory methods



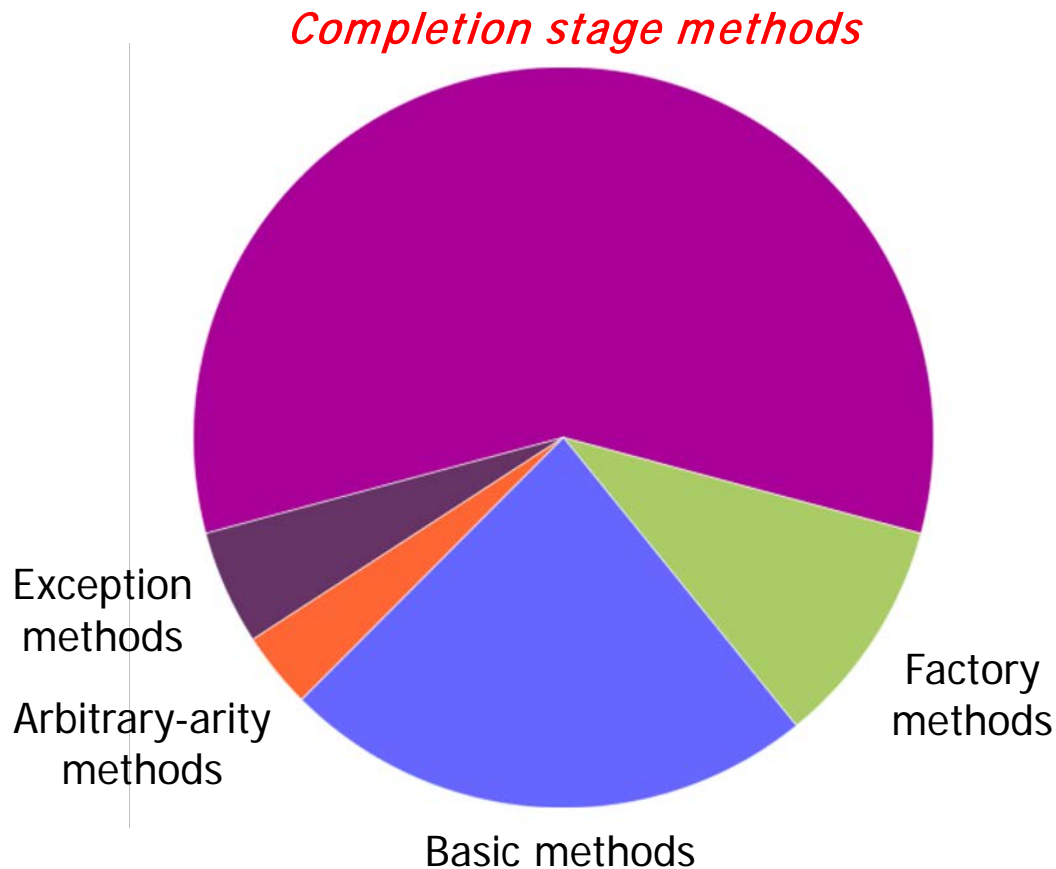
Grouping the Java 8 Completable Future API

- Other completable future features are more advanced
 - Factory methods
 - Initiate async two-way or one-way computations

<<Java Class>>	
G CompletableFuture<T>	
•	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>
•	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>

Grouping the Java 8 Completable Future API

- Other completable future features are more advanced
 - Factory methods
 - Completion stage methods



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

Grouping the Java 8 Completable Future API

- Other completable future features are more advanced
 - Factory methods
 - Completion stage methods
 - Chain together actions that perform async result processing & composition

<<Java Interface>>

CompletionStage<T>

- thenApply(Function<?>):CompletionStage<U>
- thenAccept(Consumer<?>):CompletionStage<Void>
- thenCombine(CompletionStage<?>,BiFunction<?>):CompletionStage<V>
- thenCompose(Function<?>):CompletionStage<U>
- whenComplete(BiConsumer<?>):CompletionStage<T>

<<Java Class>>

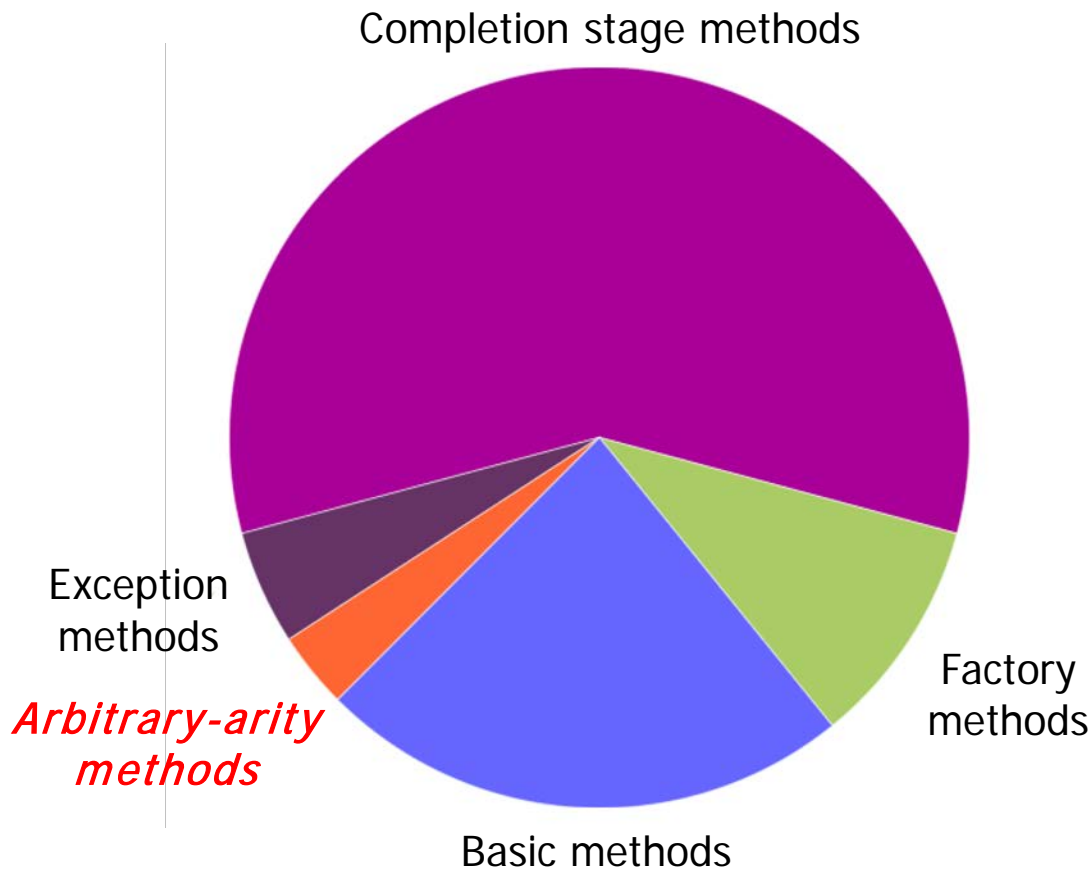
CompletableFuture<T>

- 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>

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

Grouping the Java 8 Completable Future API

- Other completable future features are more advanced
 - Factory methods
 - Completion stage methods
- “Arbitrary-arity” methods that process futures in bulk



See en.wikipedia.org/wiki/Arity

Grouping the Java 8 Completable Future API

- Other completable future features are more advanced
 - Factory methods
 - Completion stage methods
 - “Arbitrary-arity” methods that process futures in bulk
 - Combine multiple futures into a single future

<<Java Class>>	
G 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>

End of Overview of Java 8 Completable Futures (Part 2)