

Java Completable Futures: API Overview

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

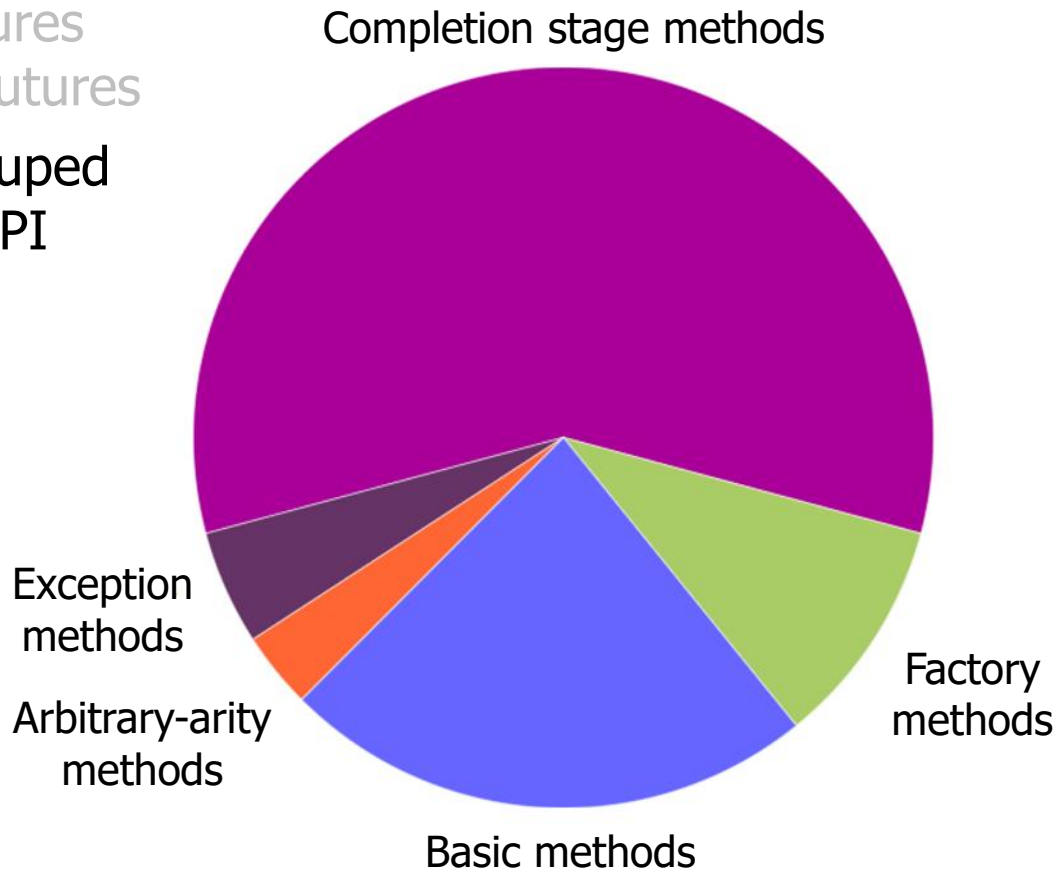
**Institute for Software
Integrated Systems**

**Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Part of the Lesson

- Know how Java completable futures overcome limitations with Java futures
- Recognize how methods are grouped in the Java completable future API



Grouping the Java Completable Future API

Grouping the Java Completable Future API

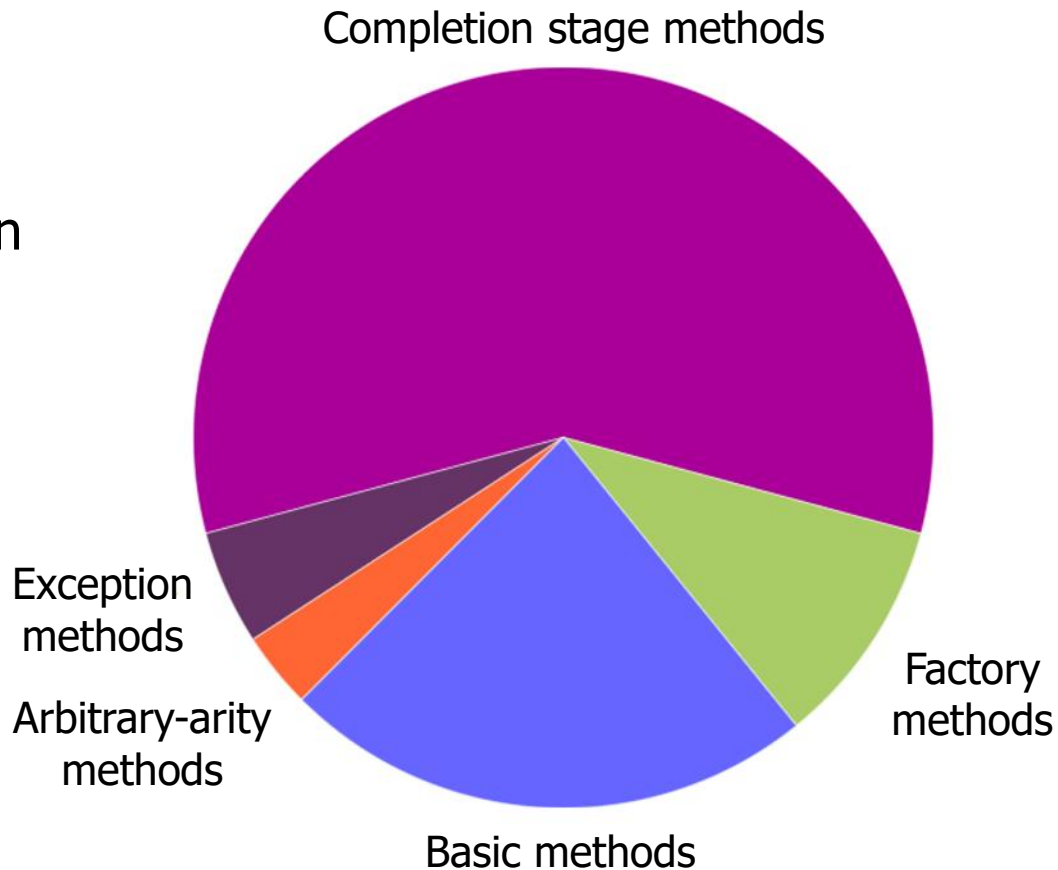
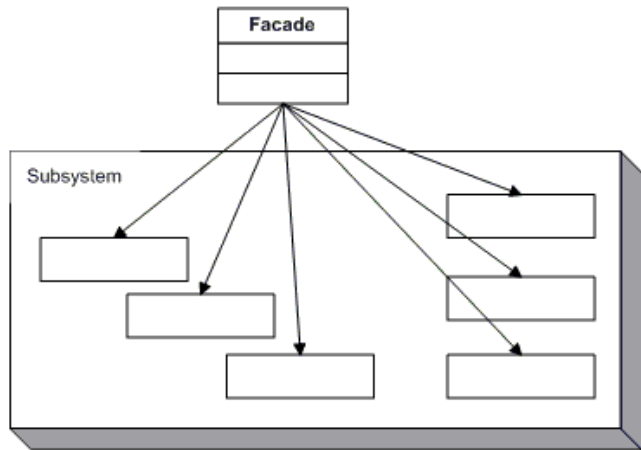
- The entire completable future framework resides in 1 public class with 60+ methods!!!

<<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 Completable Future API

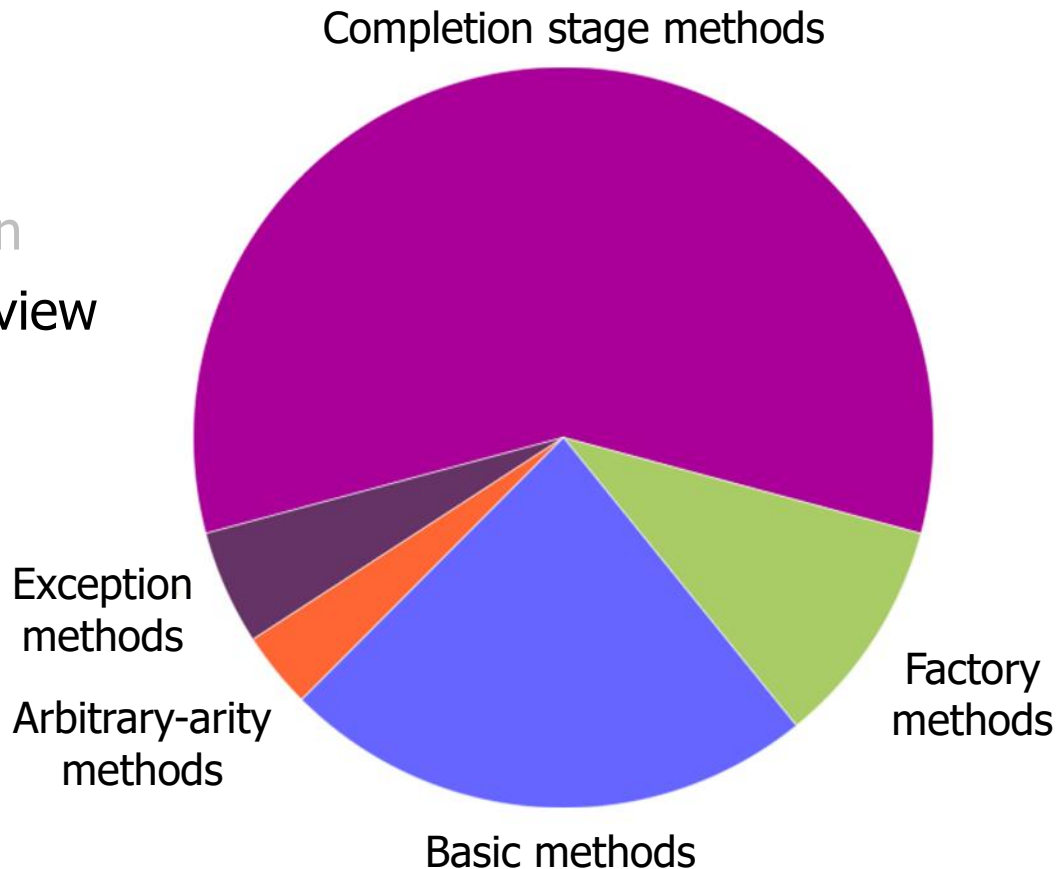
- The entire completable future framework resides in 1 public class with 60+ methods!!!
- Implements the Façade pattern



See en.wikipedia.org/wiki/Facade_pattern

Grouping the Java Completable Future API

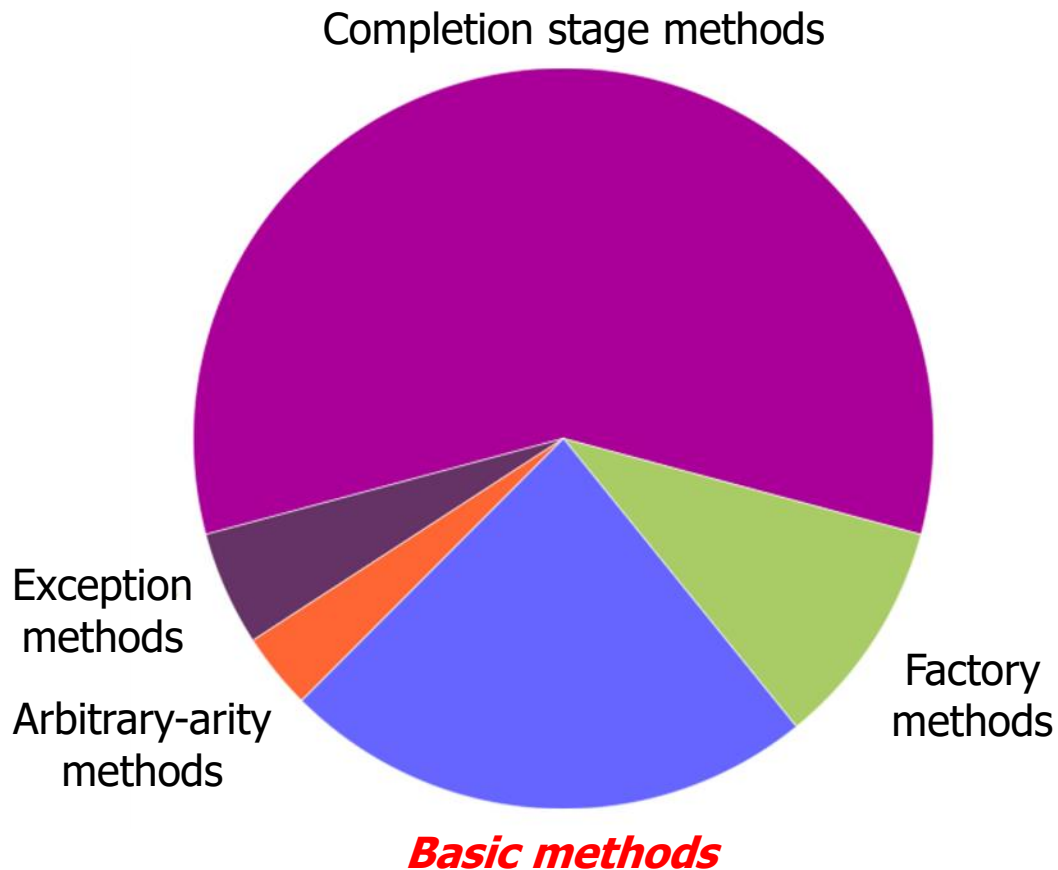
- The entire completable future framework resides in 1 public class with 60+ methods!!!
 - Implements the Façade pattern
 - It helps to have a “birds-eye” view of this method-intensive class



See en.wikipedia.org/wiki/Earthrise

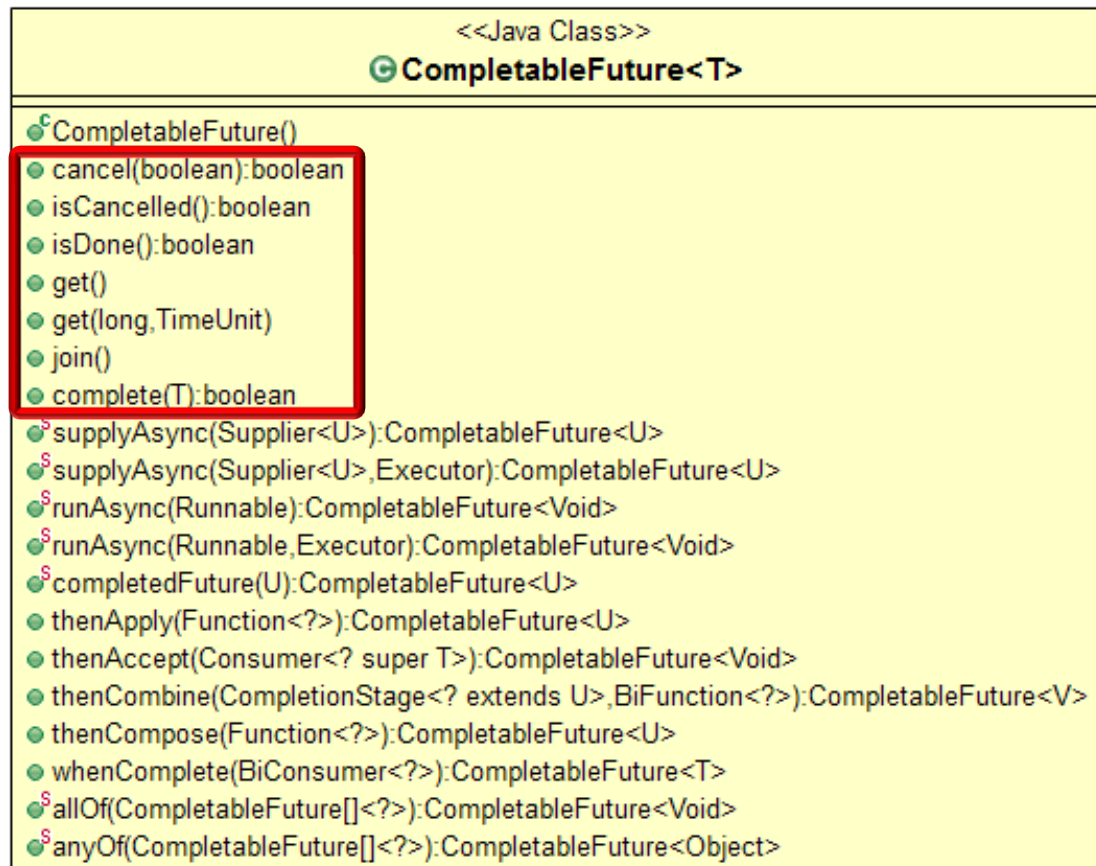
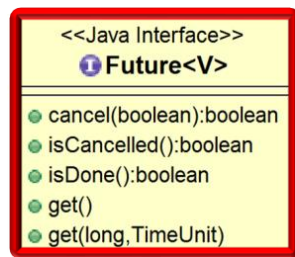
Grouping the Java Completable Future API

- Some completable future features are basic



Grouping the Java Completable Future API

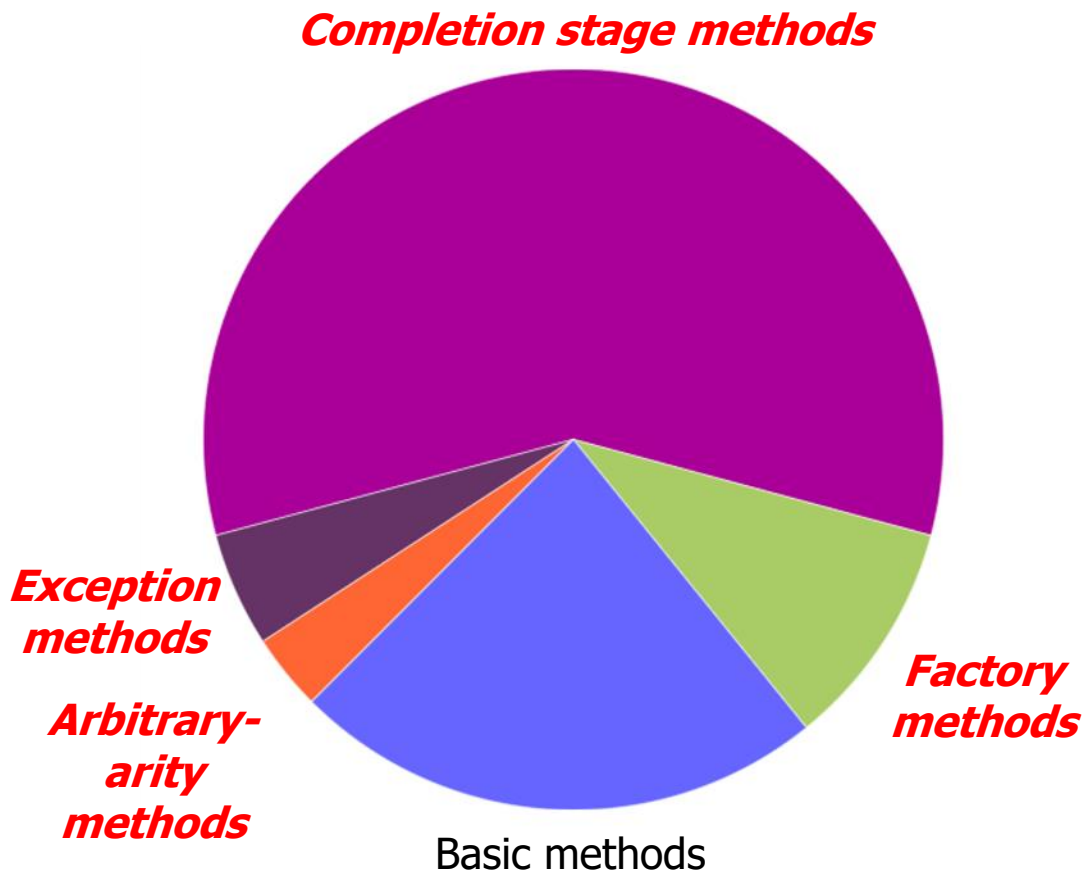
- Some completable future features are basic
- e.g., the Java Future API + some simple enhancements



Only slightly better than the conventional Future interface

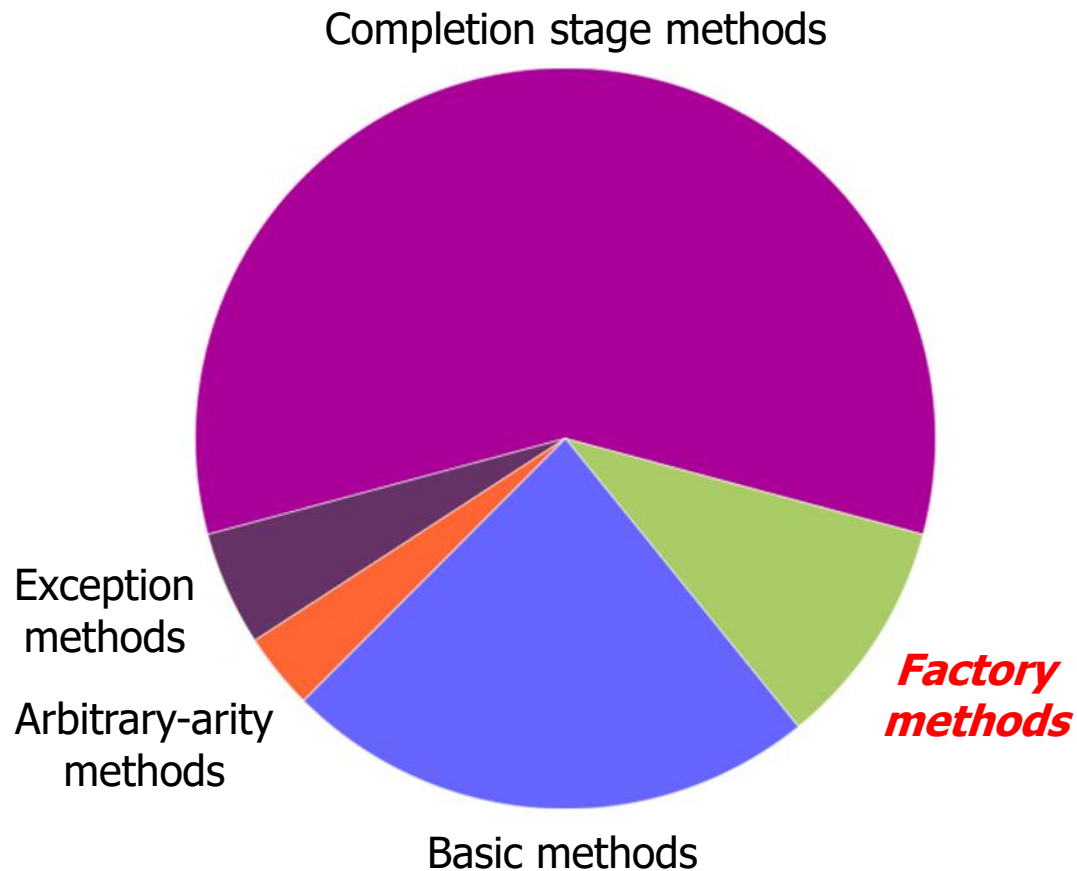
Grouping the Java Completable Future API

- Other completable future features are more advanced



Grouping the Java Completable Future API

- Other completable future features are more advanced
 - Factory methods



See en.wikipedia.org/wiki/Factory_method_pattern

Grouping the Java Completable Future API

- Other completable future features are more advanced
 - Factory methods
 - Initiate async two-way or one-way computations without using threads explicitly

<<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 Completable Future API

- Other completable future features are more advanced
- Factory methods
 - Initiate async two-way or one-way computations without using threads explicitly

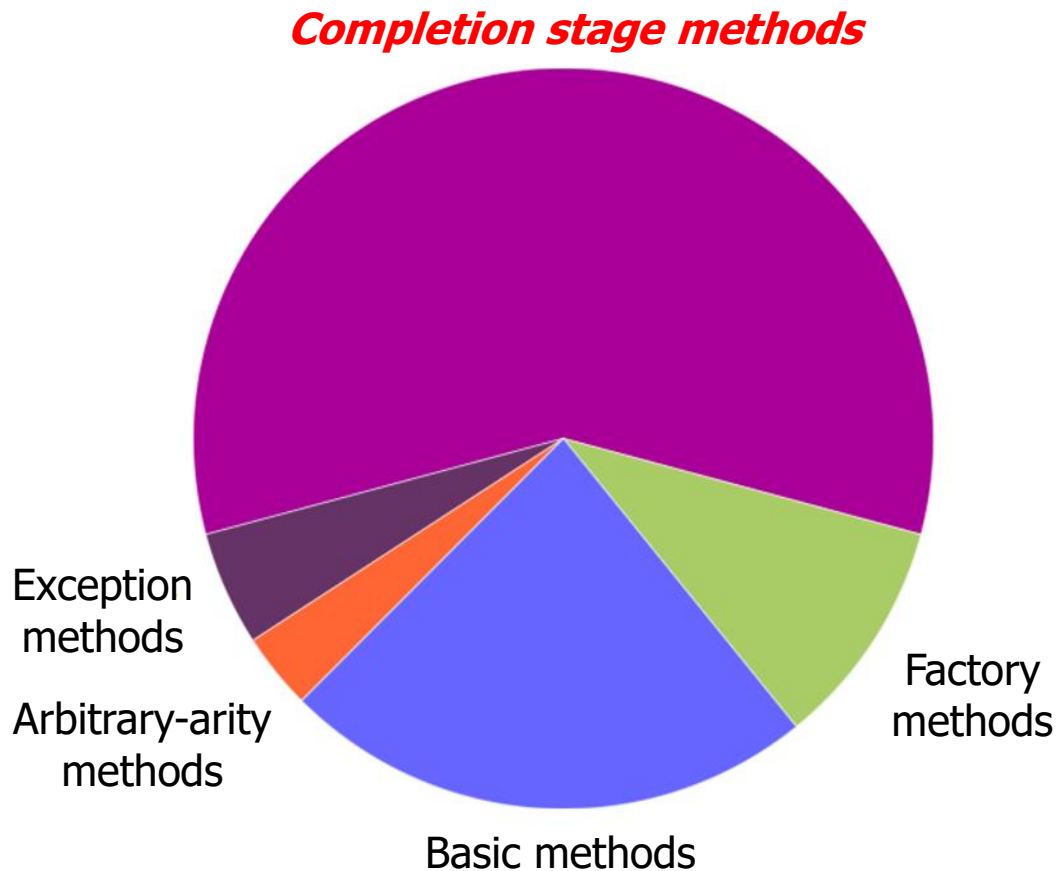


<<Java Class>>	
G CompletableFuture<T>	
•	CompletableFuture()
•	cancel(boolean):boolean
•	isCancelled():boolean
•	isDone():boolean
•	get()
•	get(long,TimeUnit)
•	join()
•	complete(T):boolean
S	supplyAsync(Supplier<U>):CompletableFuture<U>
S	supplyAsync(Supplier<U>,Executor):CompletableFuture<U>
S	runAsync(Runnable):CompletableFuture<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>

Help make programs more *elastic* by leveraging a pool of worker threads

Grouping the Java Completable Future API

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



Grouping the Java 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>>

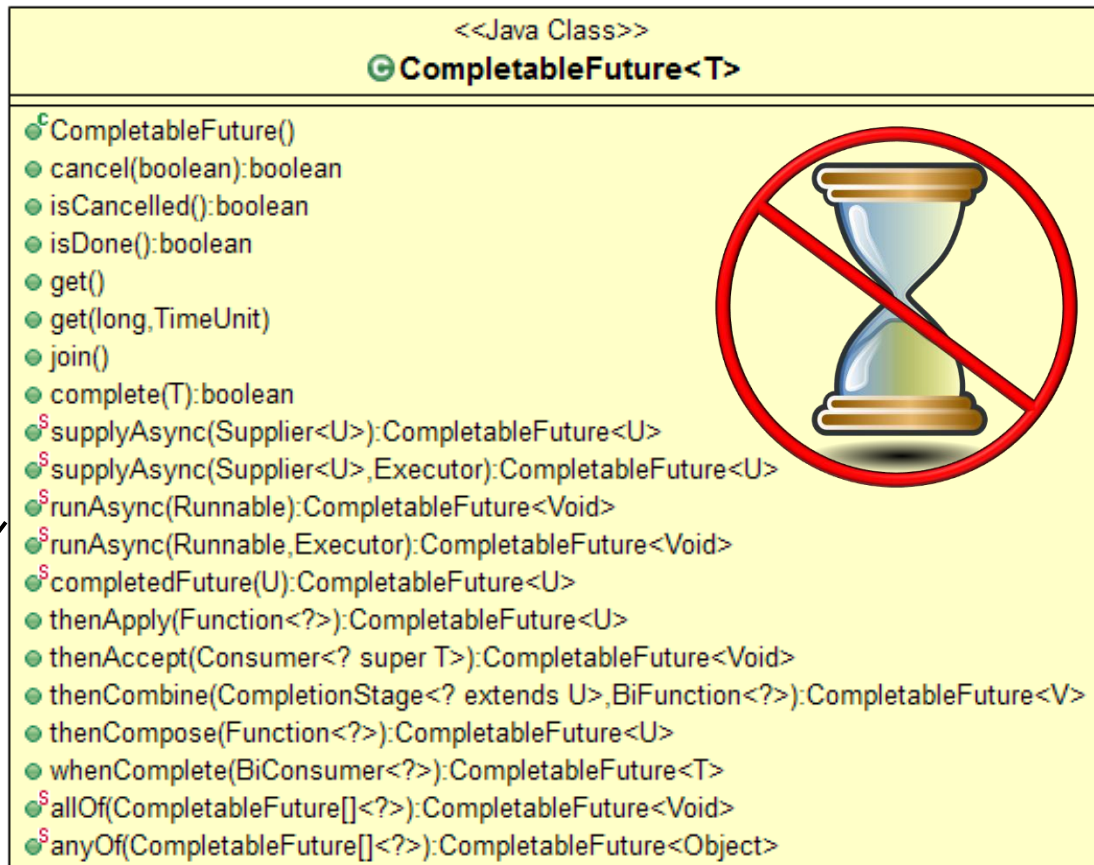
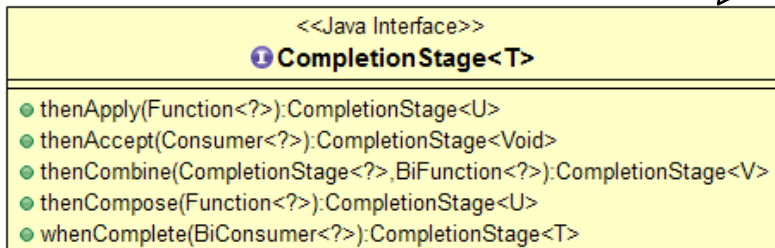
CompletableFuture<T>

```
• CompletableFuture()
• cancel(boolean):boolean
• isCancelled():boolean
• isDone():boolean
• get()
• get(long,TimeUnit)
• join()
• complete(T):boolean
• 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>
```

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

Grouping the Java Completable Future API

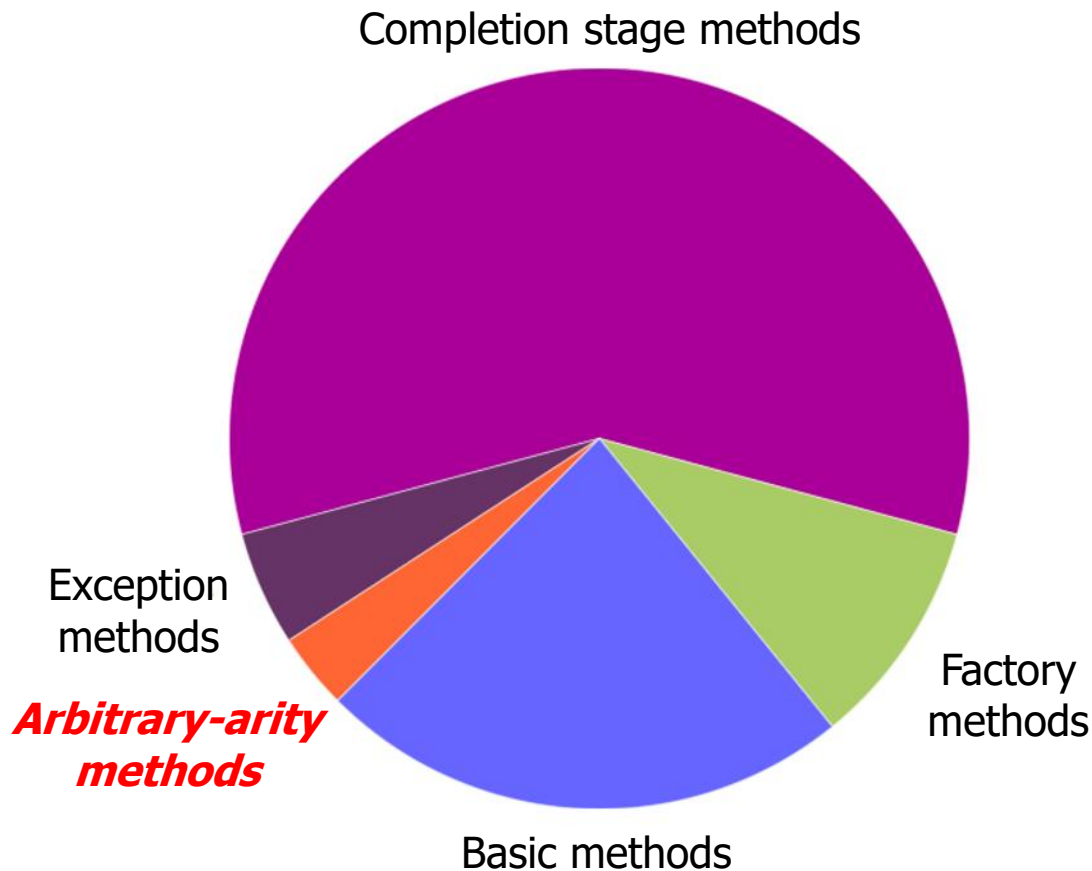
- Other completable future features are more advanced
 - Factory methods
 - Completion stage methods
 - Chain together actions that perform async result processing & composition



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

Grouping the Java Completable Future API

- Other completable future features are more advanced
 - Factory methods
 - Completion stage methods
 - “Arbitrary-arity” methods



See en.wikipedia.org/wiki/Arity

Grouping the Java Completable Future API

- Other completable future features are more advanced
 - Factory methods
 - Completion stage methods
 - “Arbitrary-arity” methods
 - Process futures in bulk by 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>

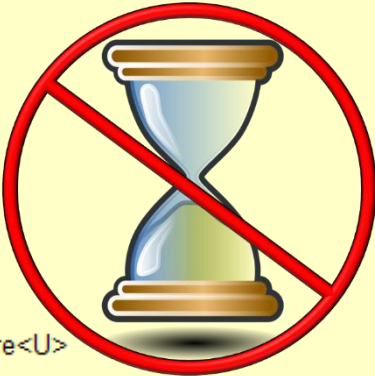
Grouping the Java Completable Future API

- Other completable future features are more advanced
 - Factory methods
 - Completion stage methods
 - “Arbitrary-arity” methods
 - Process futures in bulk by combine multiple futures into a single future

<<Java Class>>

CompletableFuture<T>

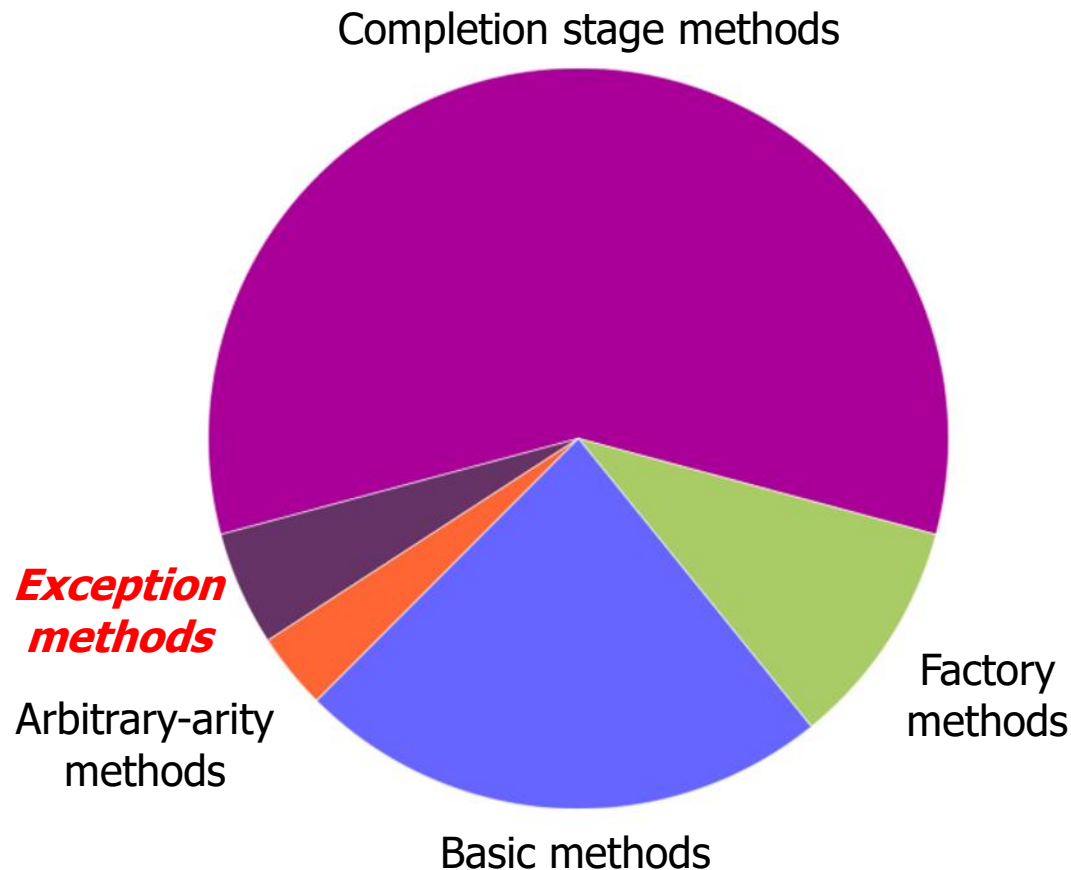
- CompletableFuture()
- cancel(boolean):boolean
- isCancelled():boolean
- isDone():boolean
- get()
- get(long,TimeUnit)
- join()
- complete(T):boolean
- ^SsupplyAsync(Supplier<U>):CompletableFuture<U>
- ^SsupplyAsync(Supplier<U>,Executor):CompletableFuture<U>
- ^SrunAsync(Runnable):CompletableFuture<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 user code

Grouping the Java Completable Future API

- Other completable future features are more advanced
 - Factory methods
 - Completion stage methods
 - “Arbitrary-arity” methods
 - Exception methods



Grouping the Java Completable Future API

- Other completable future features are more advanced
 - Factory methods
 - Completion stage methods
 - “Arbitrary-arity” methods
- Exception methods
 - Handle exceptional conditions at runtime

<<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>

Grouping the Java Completable Future API

- Other completable future features are more advanced
 - Factory methods
 - Completion stage methods
 - “Arbitrary-arity” methods
- Exception methods
 - Handle exceptional conditions at runtime



<<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>

Help make programs more *resilient* by handling erroneous computations gracefully

End of Java Completable Futures: API Overview