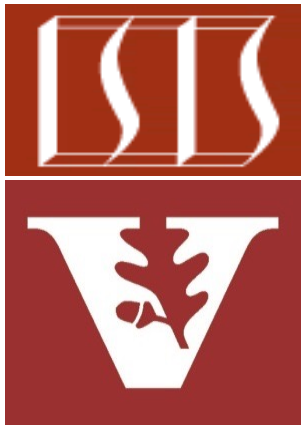
Advanced Java CompletableFuture Features: Single Stage Completion Methods (Part 1)

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt



Professor of Computer Science

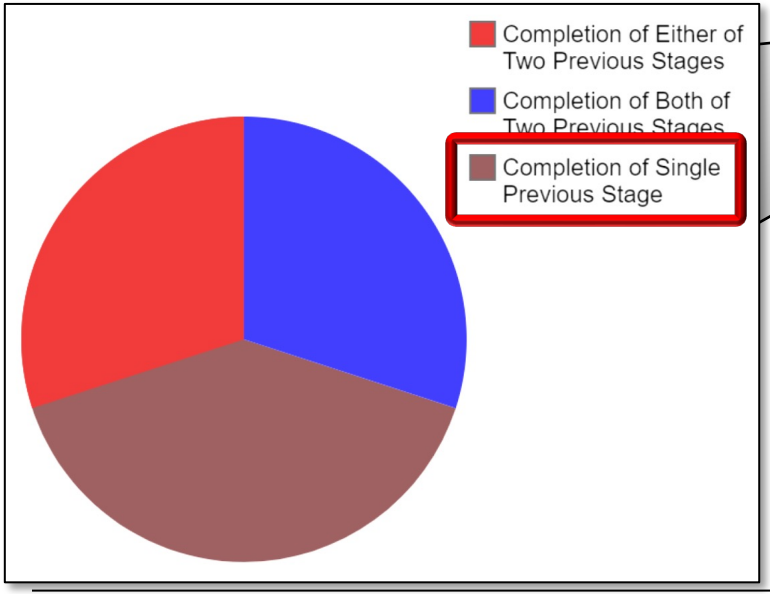
Institute for Software Integrated Systems

Vanderbilt University

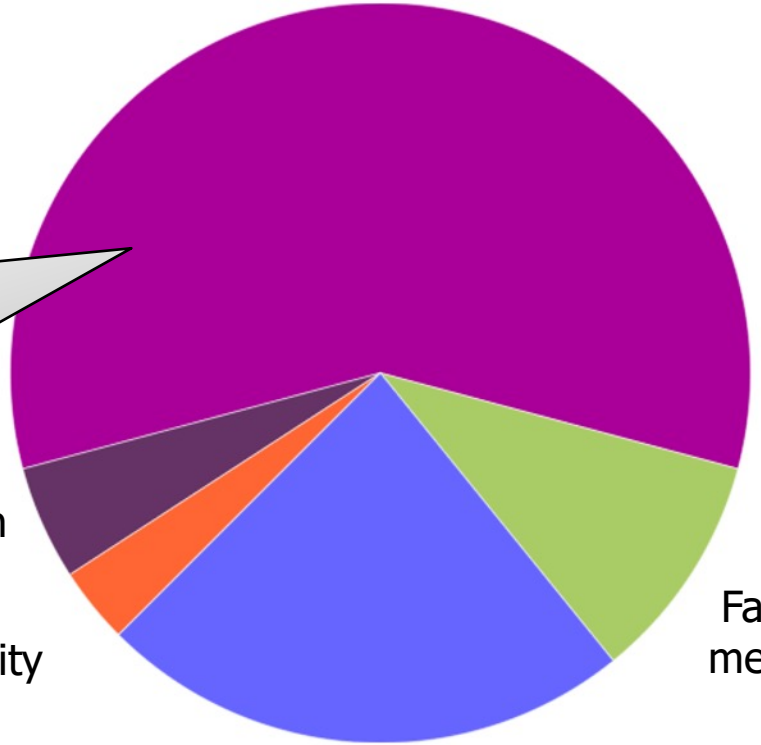
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand how completion stage methods chain dependent actions
- Know how to group these methods
- Single stage methods



Completion stage methods



Exception methods

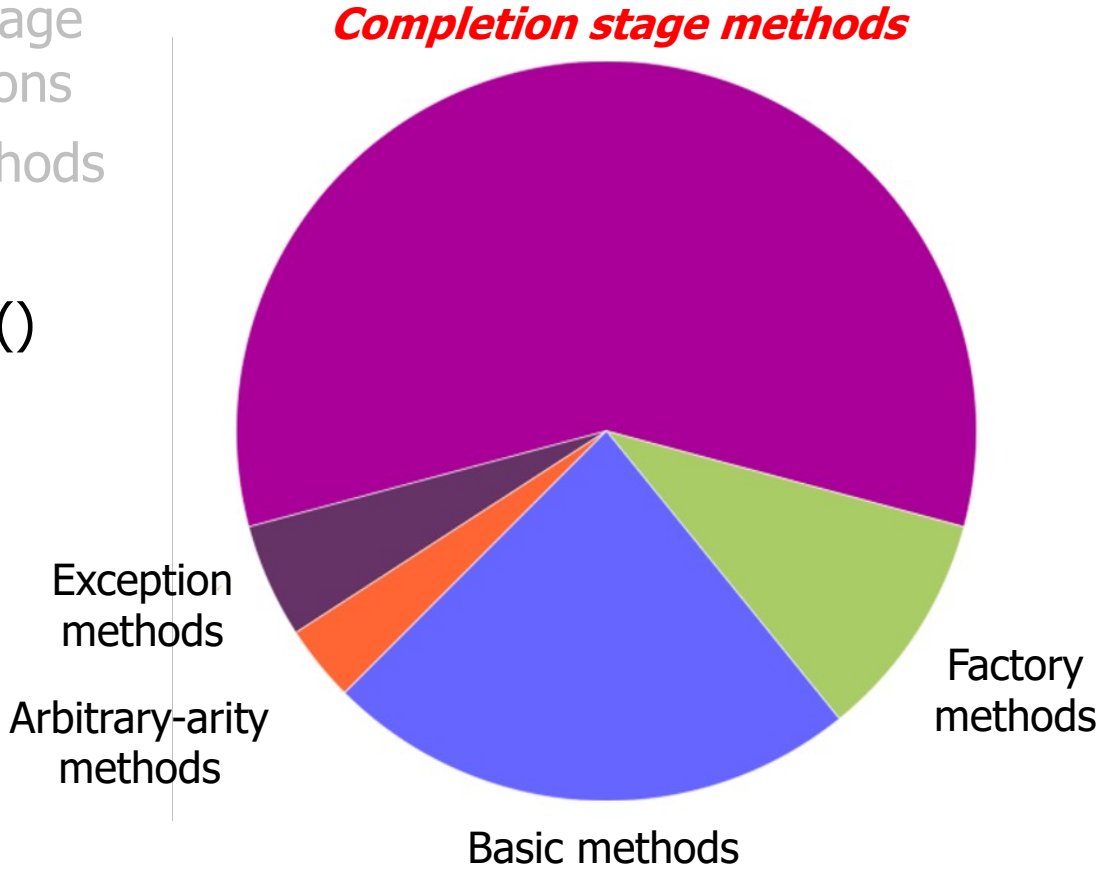
Arbitrary-arity methods

Factory methods

Basic methods

Learning Objectives in this Part of the Lesson

- Understand how completion stage methods chain dependent actions
- Know how to group these methods
- Single stage methods, e.g.
  - thenApply() & thenCompose()



Methods Triggered by Completion of a Single Stage

Methods Triggered by Completion of a Single Stage

- Methods triggered by completion of a single previous stage
  - thenApply()

```
CompletableFuture<U> thenApply
   (Function<? super T,
             ? extends U> fn)
{ ... }
```

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

Methods Triggered by Completion of a Single Stage

- Methods triggered by completion of a single previous stage
  - thenApply()
    - Applies a Function action to the previous stage's result

```
CompletableFuture<U> thenApply
   (Function<? super T,
             ? extends U> fn)
{ ... }
```

See docs.oracle.com/javase/8/docs/api/java/util/function/Function.html

Methods Triggered by Completion of a Single Stage

- Methods triggered by completion of a single previous stage
  - thenApply()
    - Applies a Function action to the previous stage's result
    - Returns a future containing the result of the action

```
CompletableFuture<U> thenApply
   (Function<? super T,
             ? extends U> fn)
{ ... }
```

Methods Triggered by Completion of a Single Stage

- Methods triggered by completion of a single previous stage
  - thenApply()
    - Applies a Function action to the previous stage's result
    - Returns a future containing the result of the action
    - Used for a quick *sync* action that returns a value rather than a future

```
BigFraction unreduced = BigFraction
   .valueOf(new BigInteger("..."),
            new BigInteger("..."),
            false); // Don't reduce!

Supplier<BigFraction> reduce = ()
  -> BigFraction.reduce(unreduced);

CompletableFuture
   .supplyAsync(reduce)
   .thenApply(BigFraction
                ::toMixedString)
   ...
```

e.g., toMixedString() returns a string value

See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex8

Methods Triggered by Completion of a Single Stage

- Methods triggered by completion of a single previous stage
  - thenCompose()

```
CompletableFuture<U> thenCompose
   (Function<? super T,
                ? extends
                CompletionStage<U>> fn)
{ ... }
```

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

Methods Triggered by Completion of a Single Stage

- Methods triggered by completion of a single previous stage
  - thenCompose()
    - Applies a Function action to the previous stage's result

```
CompletableFuture<U> thenCompose
   (Function<? super T,
             ? extends
             CompletionStage<U>> fn)
{ ... }
```

See docs.oracle.com/javase/8/docs/api/java/util/function/Function.html

Methods Triggered by Completion of a Single Stage

- Methods triggered by completion of a single previous stage
  - thenCompose()
    - Applies a Function action to the previous stage's result
    - Returns a future containing result of the action directly
      - *i.e., not* a nested future

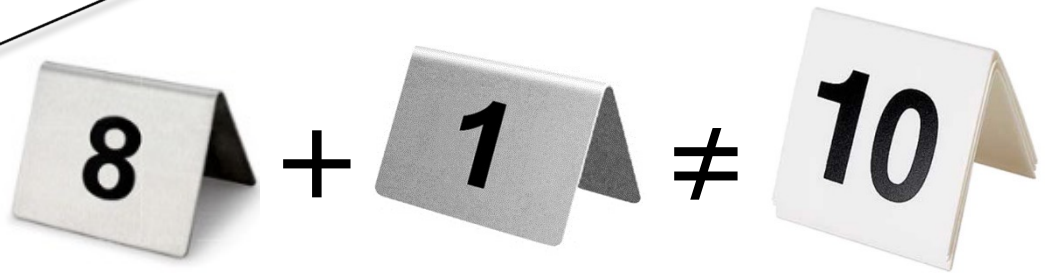
```
CompletableFuture<U> thenCompose
  (Function<? super T,
            ? extends
            CompletionStage<U>> fn)
{ ... }
```

Methods Triggered by Completion of a Single Stage

- Methods triggered by completion of a single previous stage
  - thenCompose()
    - Applies a Function action to the previous stage's result
    - Returns a future containing result of the action directly
      - *i.e., not* a nested future

```
CompletableFuture<U> thenCompose
   (Function<? super T,
             ? extends
             CompletionStage<U>> fn)
{ ... }
```

thenCompose() is similar to flatMap() on a Stream or Optional



See dzone.com/articles/understanding-flatmap

Methods Triggered by Completion of a Single Stage

- Methods triggered by completion of a single previous stage
  - thenCompose()
    - Applies a Function action to the previous stage's result
    - Returns a future containing result of the action directly
  - Used for a long-duration *async* action that returns a future

```
Function<BF,
         CompletableFuture<BF>>
  reduceAndMultiplyFractions =
    unreduced -> CompletableFuture
    .supplyAsync
      (() -> BF.reduce(unreduced))

    .thenCompose
      (reduced -> CompletableFuture
       .supplyAsync(() ->
        reduced.multiply(...)));
...
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex8

Methods Triggered by Completion of a Single Stage

- Methods triggered by completion of a single previous stage
  - thenCompose()
    - Applies a Function action to the previous stage's result
    - Returns a future containing result of the action directly
    - Used for a long-duration *async* action that returns a future

```
Function<BF,
         CompletableFuture<BF>>
reduceAndMultiplyFractions =
  unreduced -> CompletableFuture
  .supplyAsync
    (() -> BF.reduce(unreduced))
```

This function reduces & multiplies big fractions

```
  .thenCompose
    (reduced -> CompletableFuture
     .supplyAsync(() ->
      reduced.multiply(...)));
...
```

See docs.oracle.com/javase/8/docs/api/java/util/function/Function.html

Methods Triggered by Completion of a Single Stage

- Methods triggered by completion of a single previous stage
  - thenCompose()
    - Applies a Function action to the previous stage's result
    - Returns a future containing result of the action directly
  - Used for a long-duration *async* action that returns a future

```
Function<BF,
         CompletableFuture<BF>>
  reduceAndMultiplyFractions =
    unreduced -> CompletableFuture
    .supplyAsync
      (() -> BF.reduce(unreduced))
    .thenCompose
      (reduced -> CompletableFuture
       .supplyAsync(() ->
        reduced.multiply(...)));
...
```

Reduce big fraction asynchronously & return a completable future

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

Methods Triggered by Completion of a Single Stage

- Methods triggered by completion of a single previous stage
  - thenCompose()
    - Applies a Function action to the previous stage's result
    - Returns a future containing result of the action directly
    - Used for a long-duration *async* action that returns a future

```
Function<BF,
         CompletableFuture<BF>>
  reduceAndMultiplyFractions =
    unreduced -> CompletableFuture
    .supplyAsync
      (() -> BF.reduce(unreduced))
```

```
    .thenCompose
      (reduced -> CompletableFuture
       .supplyAsync(() ->
        reduced.multiply(...)));
...
```

supplyAsync() returns a future, but thenCompose() "flattens" this future

See docs.orade.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#supplyAsync

End of Advanced Java CompletableFuture Features: Single Stage Completion Methods (Part 1)