

Advanced Java CompletableFuture Features: Handling Runtime Exceptions (Part 1)

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt



Professor of Computer Science

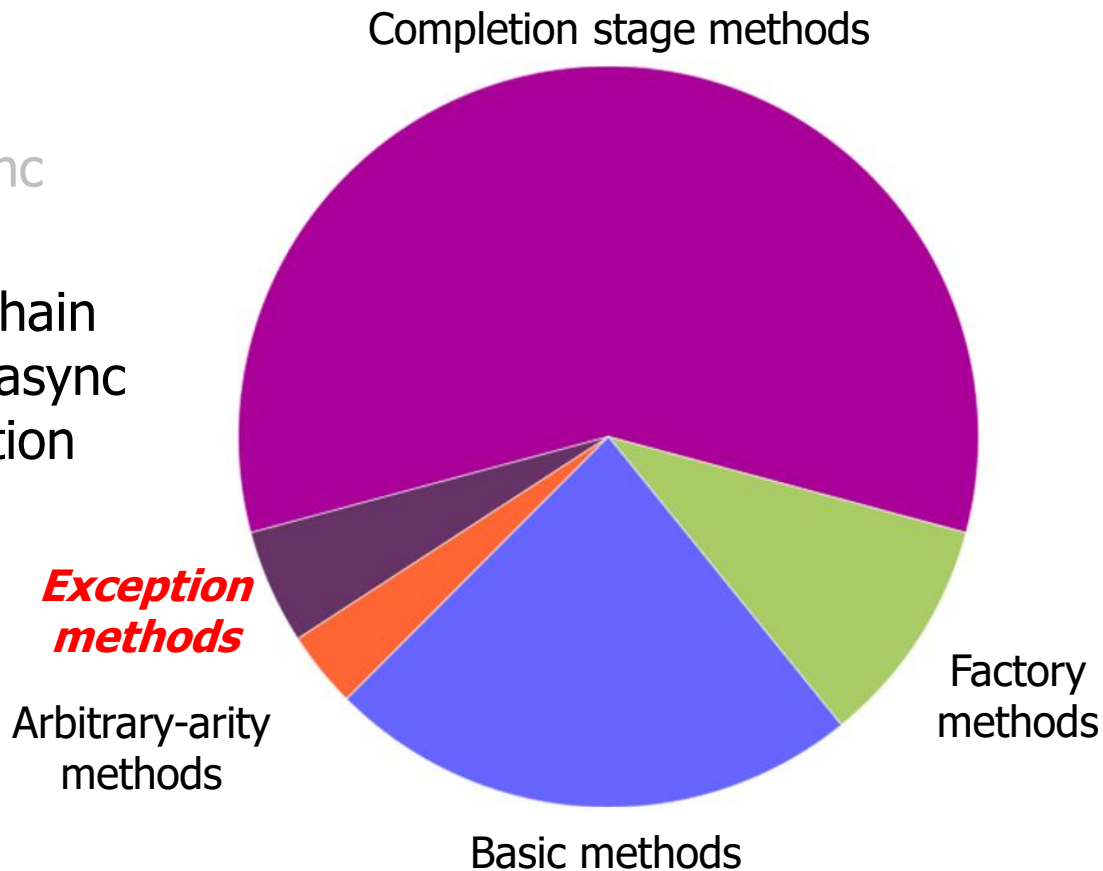
**Institute for Software
Integrated Systems**

**Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Part of the Lesson

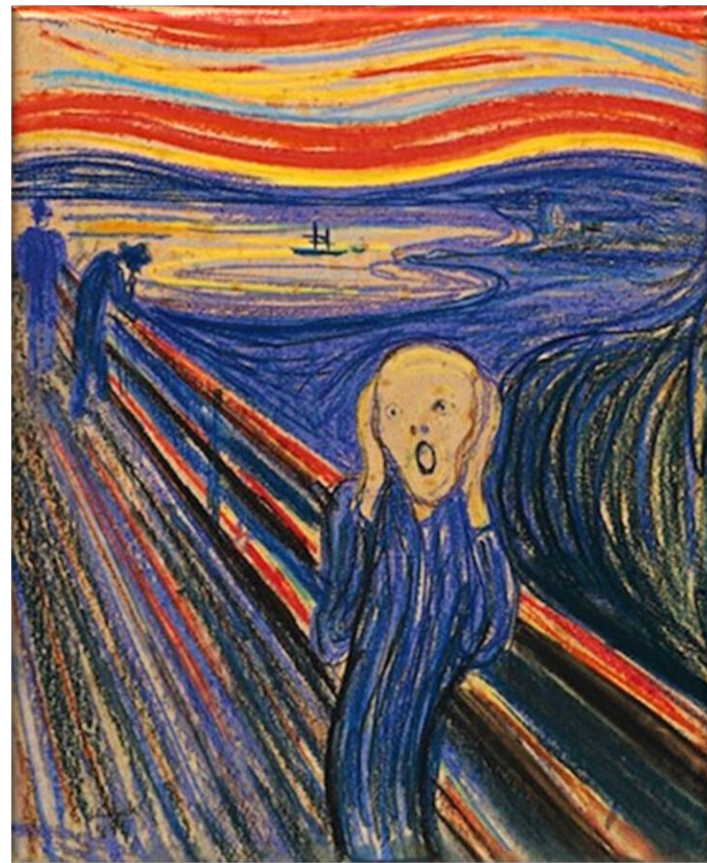
- Understand advanced features of completable futures, e.g.
 - Factory methods initiate async computations
- Completion stage methods chain together actions to perform async result processing & composition
 - Method grouping
 - Single stage methods
 - Two stage methods (and)
 - Two stage methods (or)
 - Apply these methods
- Handle runtime exceptions (Overview)



Overview of Handling Exceptions in Completion Stages

Overview of Handling Exceptions in Completion Stages

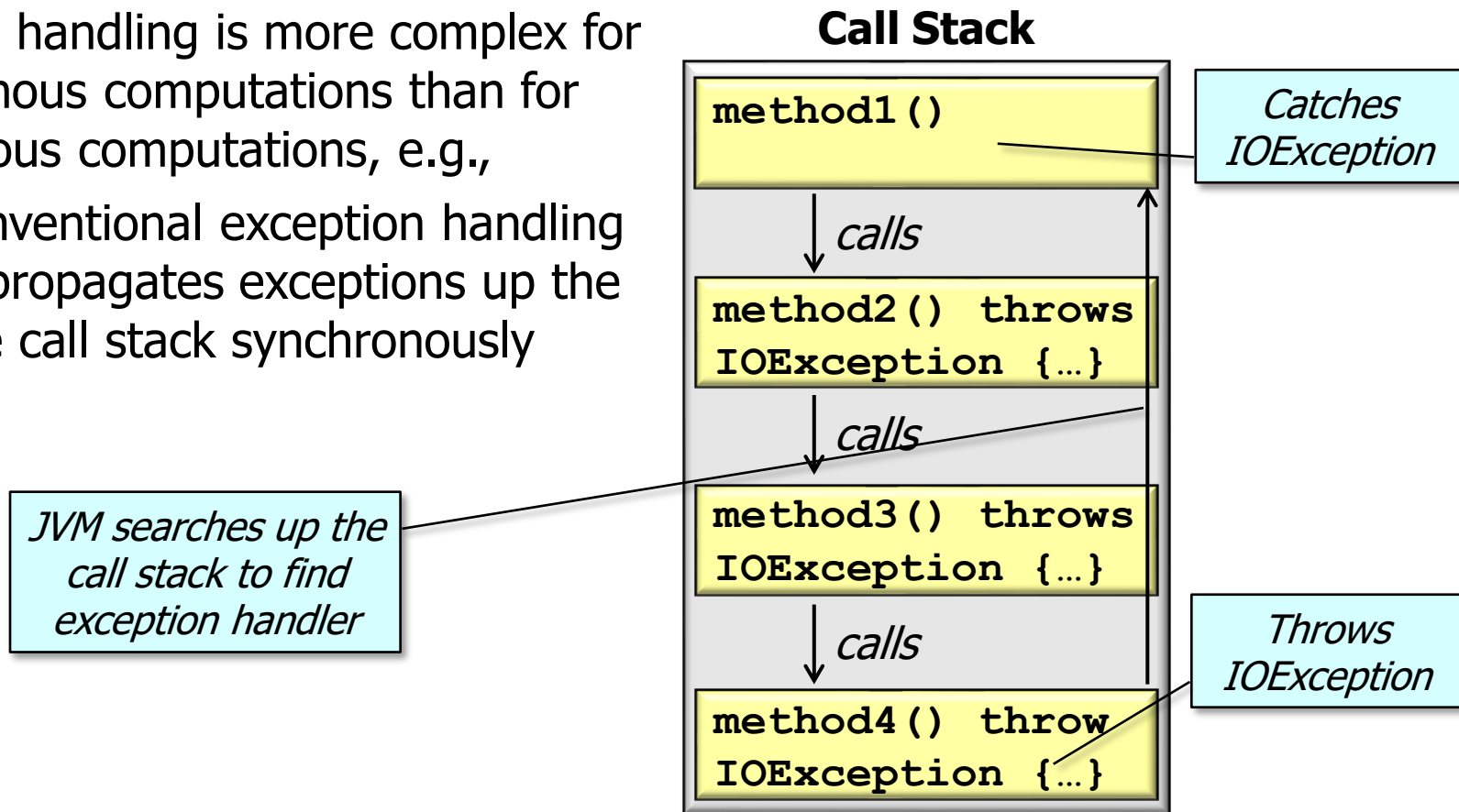
- Exception handling is more complex for asynchronous computations than for synchronous computations



See blog.lightstreamer.com/2014/07/exception-handling-in-asynchronous-java.html

Overview of Handling Exceptions in Completion Stages

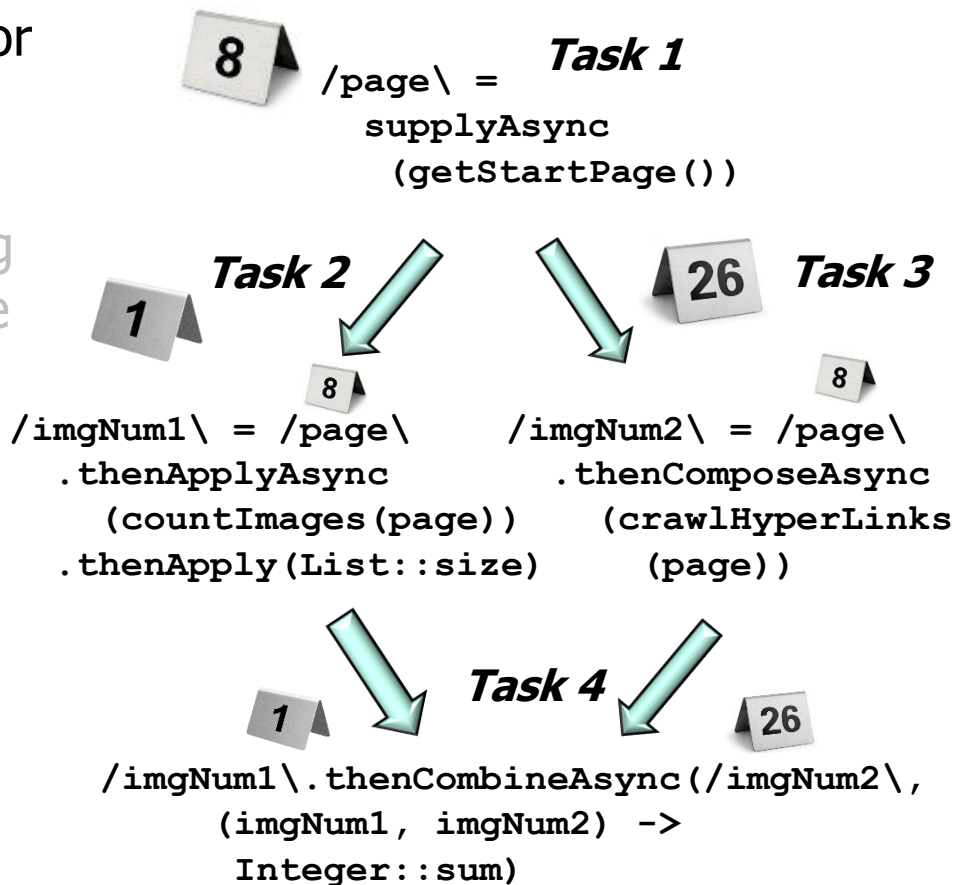
- Exception handling is more complex for asynchronous computations than for synchronous computations, e.g.,
- The conventional exception handling model propagates exceptions up the runtime call stack synchronously



See en.wikipedia.org/wiki/Exception_handling

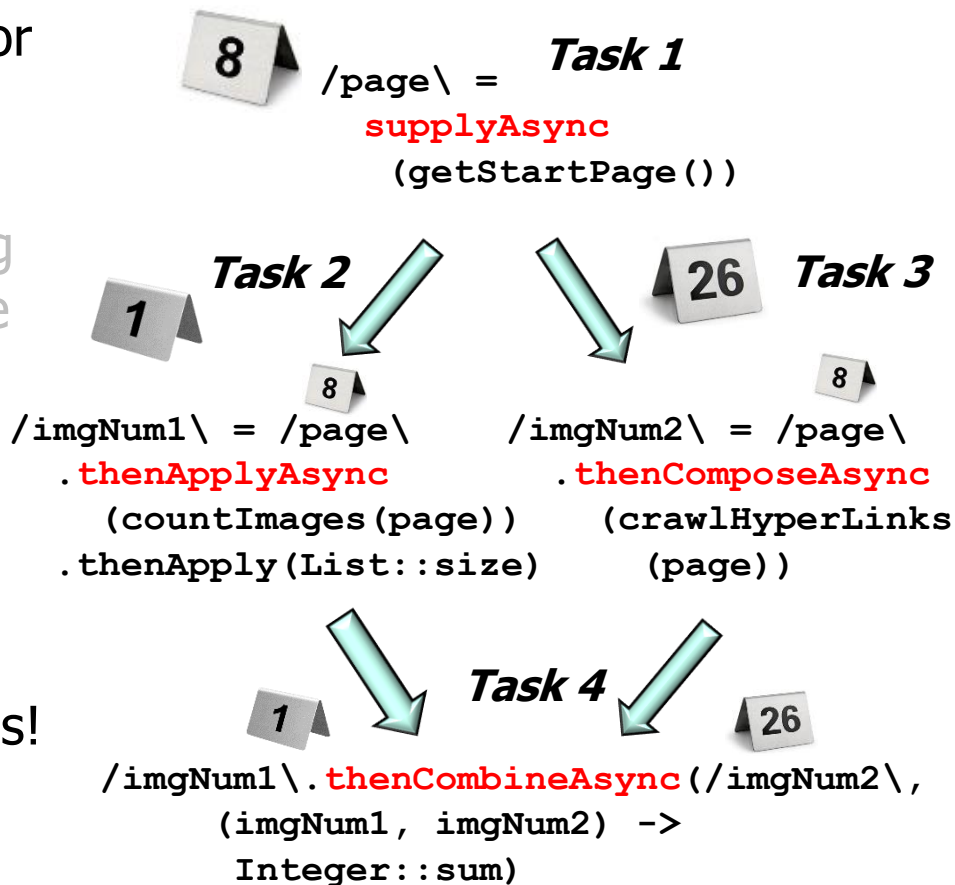
Overview of Handling Exceptions in Completion Stages

- Exception handling is more complex for asynchronous computations than for synchronous computations, e.g.,
 - The conventional exception handling model propagates exceptions up the runtime call stack synchronously
- However, completable futures that run asynchronously don't conform to a conventional call stack model



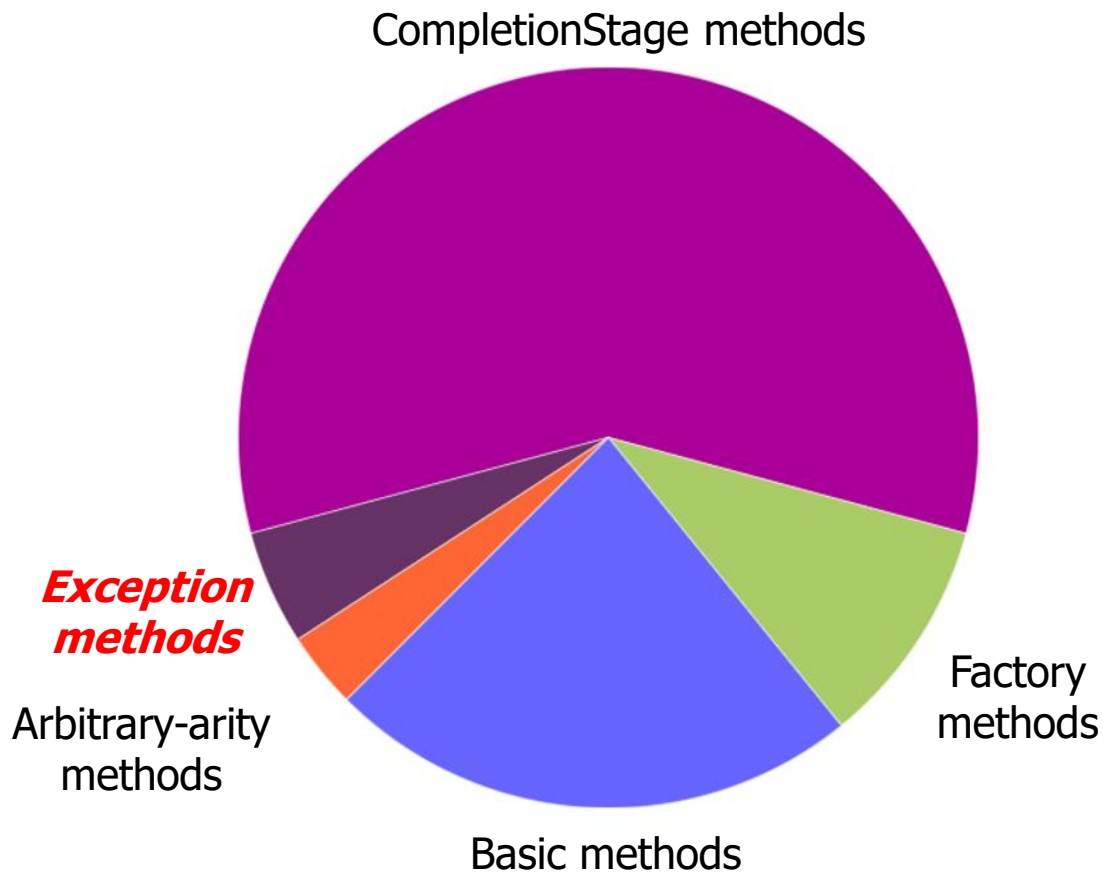
Overview of Handling Exceptions in Completion Stages

- Exception handling is more complex for asynchronous computations than for synchronous computations, e.g.,
 - The conventional exception handling model propagates exceptions up the runtime call stack synchronously
- However, completable futures that run asynchronously don't conform to a conventional call stack model
 - e.g., completion stage methods can run in different worker threads!



Overview of Handling Exceptions in Completion Stages

- Completion stage methods handle runtime exceptions that occur asynchronously



See www.jesperdj.com/2015/09/26/the-future-is-completable-in-java-8

Overview of Handling Exceptions in Completion Stages

- Completion stage methods handle runtime exceptions that occur asynchronously

Methods	Params	Returns	Behavior
<code>whenComplete(Async)</code>	<code>BiConsumer</code>	<code>CompletableFuture</code> with result of earlier stage or throws exception	Handle outcome of a stage, whether a result value or an exception
<code>handle(Async)</code>	<code>BiFunction</code>	<code>CompletableFuture</code> with result of <code>BiFunction</code>	Handle outcome of a stage & return new value
<code>exceptionally</code>	<code>Function</code>	<code>CompletableFuture<T></code>	When exception occurs, replace exception with result value

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

Overview of Handling Exceptions in Completion Stages

- Completion stage methods handle runtime exceptions that occur asynchronously

Methods	Params	Returns	Behavior
<code>whenComplete(Async)</code>	<code>BiConsumer</code>	<code>CompletableFuture</code> with result of earlier stage or throws exception	Handle outcome of a stage, whether a result value or an exception
<code>handle(Async)</code>	<code>BiFunction</code>	<code>CompletableFuture</code> with result of <code>BiFunction</code>	Handle outcome of a stage & return new value
<code>exceptionally(Async)</code>	<code>Function</code>	<code>CompletableFuture<T></code>	When exception occurs, replace exception with result value

Added in Java 12

End of Advanced Java CompletableFuture Features: Handling Runtime Exceptions (Part 1)