Advanced Java Completable Future Features: Handling Runtime Exceptions (Part 1)

Douglas C. Schmidt

<u>d.schmidt@vanderbilt.edu</u>

www.dre.vanderbilt.edu/~schmidt



Professor of Computer Science

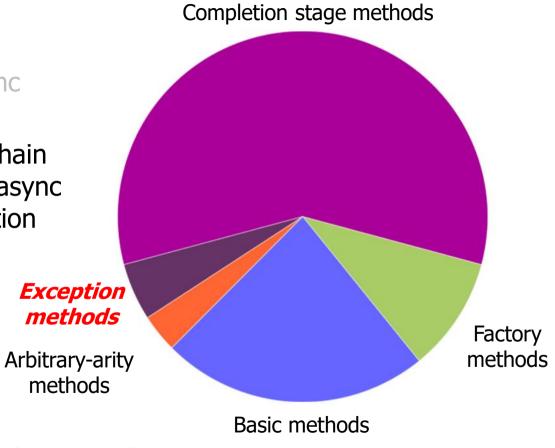
Institute for Software Integrated Systems

Vanderbilt University Nashville, Tennessee, USA

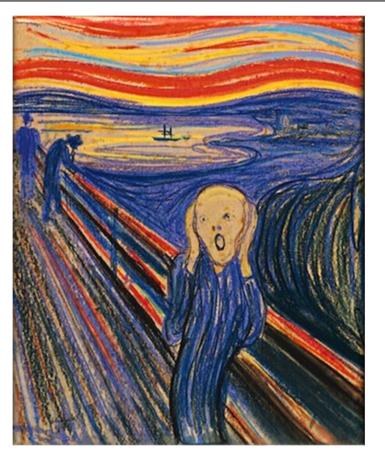


Learning Objectives in this Part of the Lesson

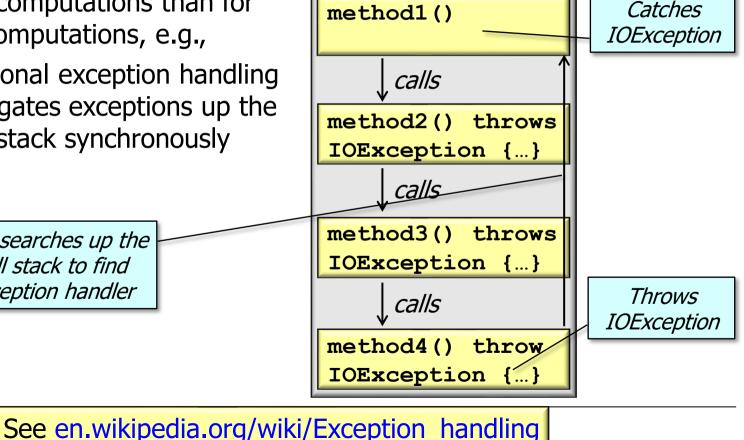
- Understand advanced features of completable futures, e.g.
 - Factory methods initiate async computations
 - Completion stage methods chain together actions to perform async result processing & composition
 - Method grouping
 - Single stage methods
 - Two stage methods (and)
 - Two stage methods (or)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



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



- 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



Call Stack

JVM searches up the call stack to find exception handler

- 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

```
supplyAsync
                 (getStartPage())
         Task 2
/imgNum1 = /page
                      /imgNum2 = /page
  .thenApplyAsync
                         .thenComposeAsync
    (countImages (page))
                           (crawlHyperLinks
  .thenApply(List::size)
                             (page))
                    Task 4
```

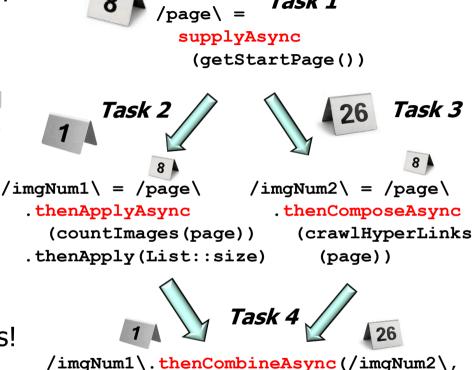
/imgNum1\.thenCombineAsync(/imgNum2\,

(imgNum1, imgNum2) ->

Integer::sum)



- 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!

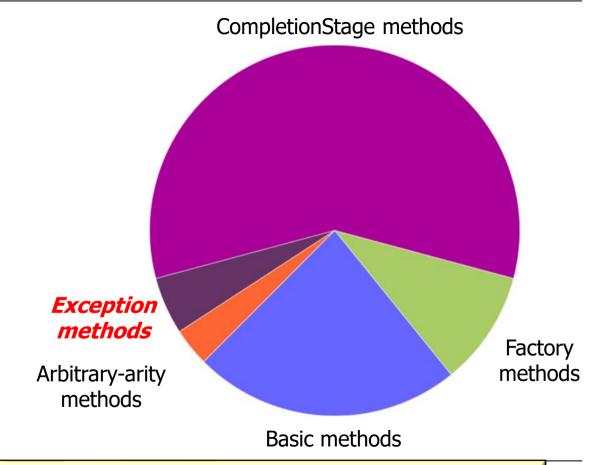


(imgNum1, imgNum2) ->

Integer::sum)

See suryanarayanjena.wordpress.com/async-methods-in-completable-future

 Completion stage methods handle runtime exceptions that occur asynchronously



Completion stage methods handle runtime exceptions that occur asynchronously

 Methods Particular when Complete Complete (Async)

 handle

thods Params Returns Behavior

Then Bi Completable Handle outcome

The consumer Future with of a stage,

The result of whether a result

The earlier stage value or an or throws exception

exception Completable Handle outcome handle Bi Function Future with (Async) of a stage & return new value result of **BiFunction** exceptionally Function Completable When exception Future<T> occurs, replace exception with

result value

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

 Completion stage methods **Methods Params** Returns **Behavior** handle runtime exceptions when Bi Completable Handle outcome that occur asynchronously **Consumer Future with** Complete of a stage, (Async) whether a result result of earlier stage value or an

or throws exception exception Completable Handle outcome handle Bi Function Future with (Async) of a stage & return new value result of **BiFunction** Added in Java 12 exceptionally Function Completable When exception Future<T> (Async) occurs, replace exception with

result value See www.logicbig.com/tutorials/core-java-tutorial/java-12-changes/completion-stage-new-methods.html

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