

The Structure & Functionality of the Java Completable Futures Framework

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

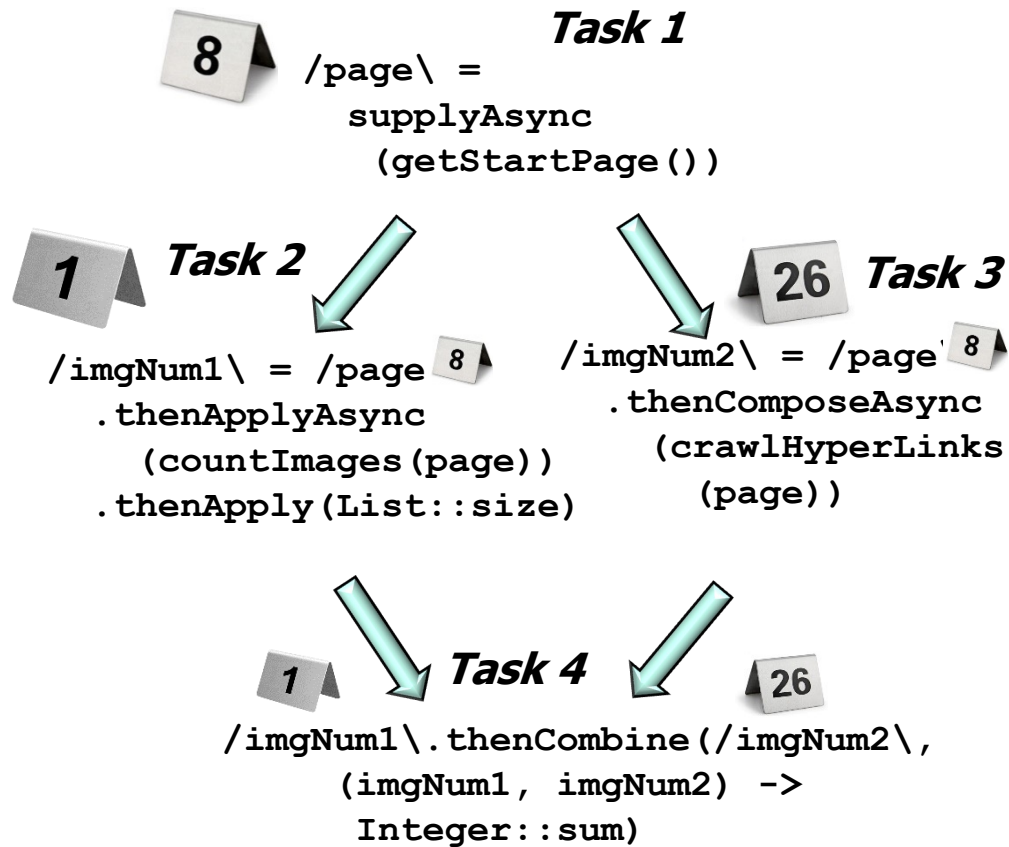
**Institute for Software
Integrated Systems**

**Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Part of the Lesson

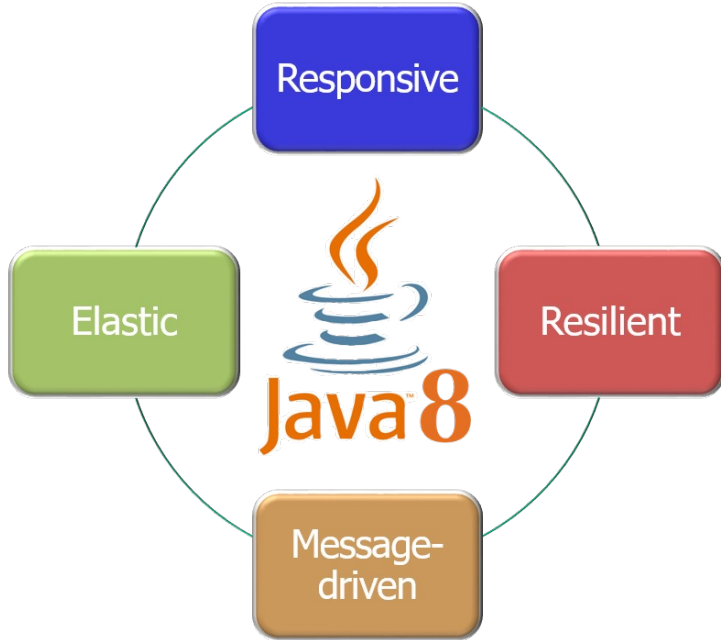
- Understand the key principles underlying reactive programming
- Recognize the Java completable futures framework's structure & functionality



Overview of the Java Completable Futures Framework

Overview of the Java Completable Futures Framework

- Java's completable futures framework provides an asynchronous & reactive parallel programming model



Class `CompletableFuture<T>`

```
java.lang.Object  
    java.util.concurrent.CompletableFuture<T>
```

All Implemented Interfaces:

```
CompletionStage<T>, Future<T>
```

```
public class CompletableFuture<T>  
    extends Object  
    implements Future<T>, CompletionStage<T>
```

A `Future` that may be explicitly completed (setting its value and status), and may be used as a `CompletionStage`, supporting dependent functions and actions that trigger upon its completion.

When two or more threads attempt to complete, `completeExceptionally`, or cancel a `CompletableFuture`, only one of them succeeds.

In addition to these and related methods for directly manipulating status and results, `CompletableFuture` implements interface `CompletionStage` with the following policies:

Overview of the Java Completable Futures Framework

- Java's completable futures framework provides an asynchronous & reactive parallel programming model
 - As a baseline, consider a web crawler implementation that's synchronous

Step 1: *Get start page*

Step 2: *Count images on the page*

Step 3: *Count images on all hyperlinked pages*

Step 4: *Combine results to create the total*



See en.wikipedia.org/wiki/Web_crawler

Overview of the Java Completable Futures Framework

- Java's completable futures framework provides an asynchronous & reactive parallel programming model
- As a baseline, consider a web crawler implementation that's synchronous
 - The time needed to perform all these steps is the sum of each step sequentially



Step 1: Get start page

Step 2: Count images on the page

Step 3: Count images on all hyperlinked pages

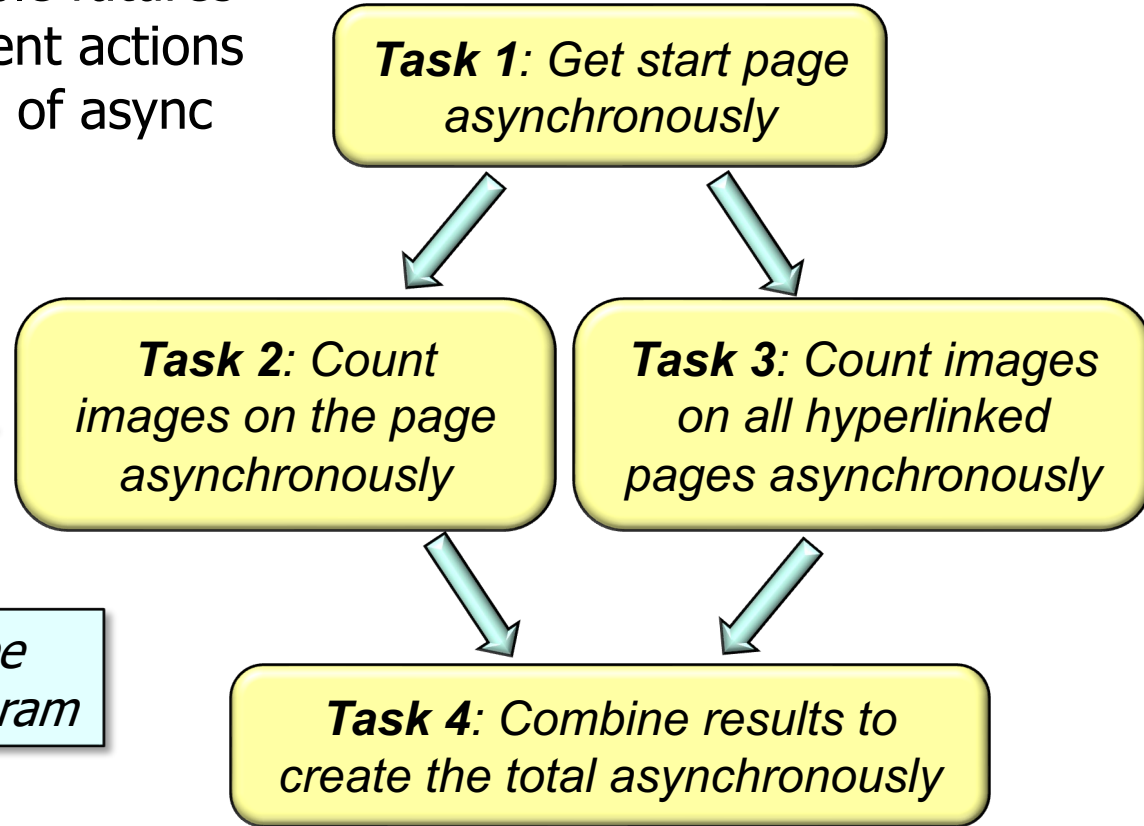
Step 4: Combine results to create the total



See en.wikipedia.org/wiki/Sequential_algorithm

Overview of the Java Completable Futures Framework

- In contrast, Java's completable futures framework supports dependent actions that trigger upon completion of async operations

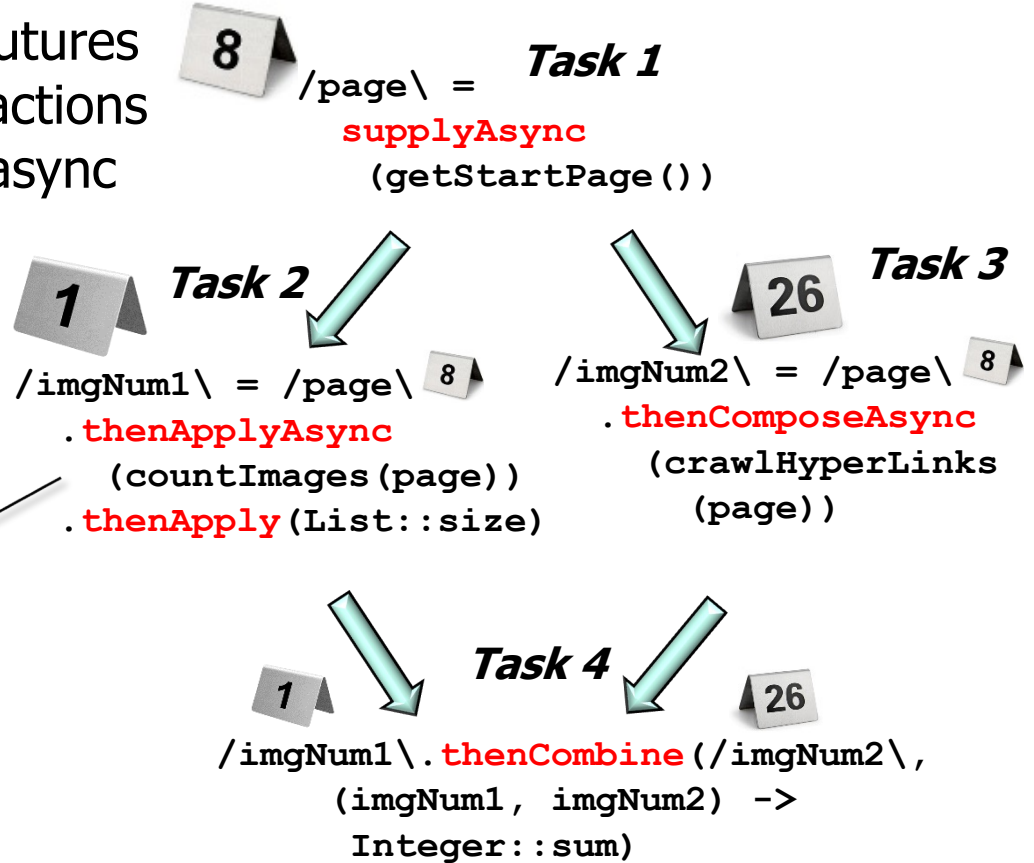


These dependencies can be modeled via a data flow diagram

See en.wikipedia.org/wiki/Data-flow_diagram

Overview of the Java Completable Futures Framework

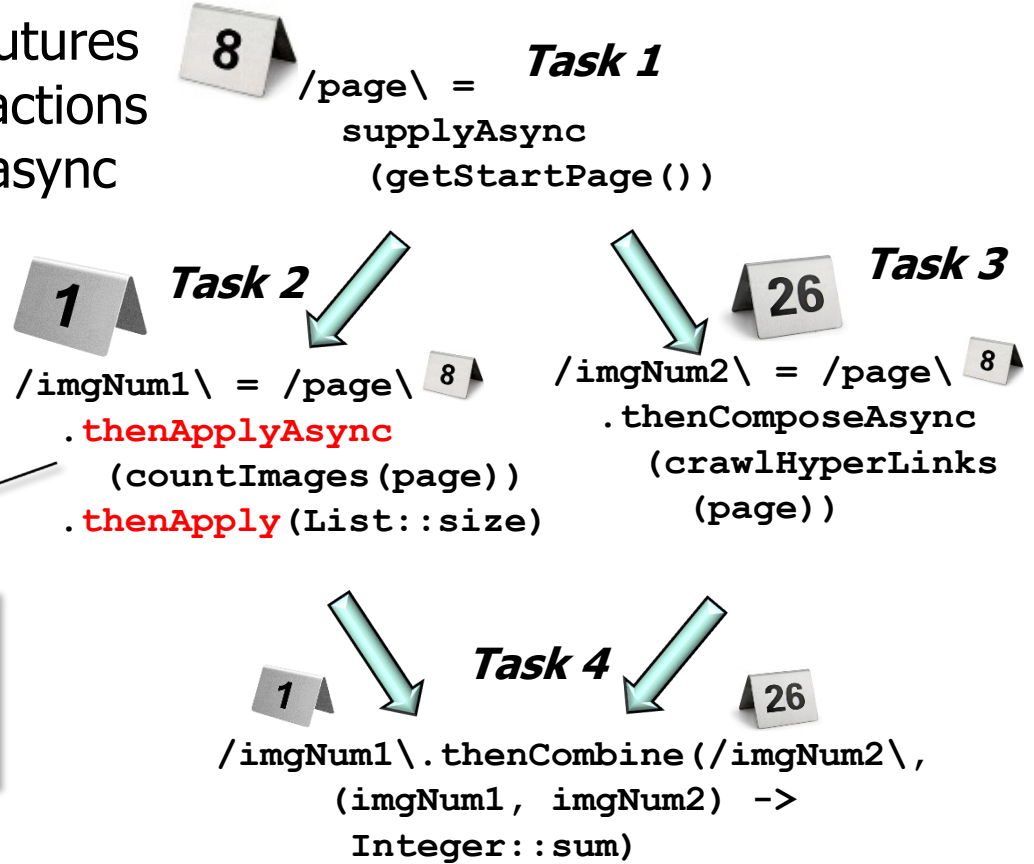
- In contrast, Java's completable futures framework supports dependent actions that trigger upon completion of async operations



Async operations can be forked, chained, & joined

Overview of the Java Completable Futures Framework

- In contrast, Java's completable futures framework supports dependent actions that trigger upon completion of async operations



Enables async programming to resemble sync programming via "completion stages"

Overview of the Java Completable Futures Framework

- In contrast, Java's completable futures framework supports dependent actions that trigger upon completion of async operations
- These async operations can run in a thread pool

8 /page\ = **Task 1**
supplyAsync
(getStartPage())



1 **Task 2** /imgNum1\ = /page\
.thenApplyAsync
(countImages(page))
.thenApply(List::size)


26 **Task 3** /imgNum2\ = /page\
.thenComposeAsync
(crawlHyperLinks
(page))

1 **Task 4** /imgNum1\.thenCombine(/imgNum2\
(imgNum1, imgNum2) ->
Integer::sum)

Overview of the Java Completable Futures Framework


- In contrast, Java's completable futures framework supports dependent actions that trigger upon completion of async operations
- These async operations can run in a thread pool
 - Either a (common) fork-join pool or various types of pre- or user-defined thread pools

8 /page\ = **Task 1**
supplyAsync
(getStartPage())

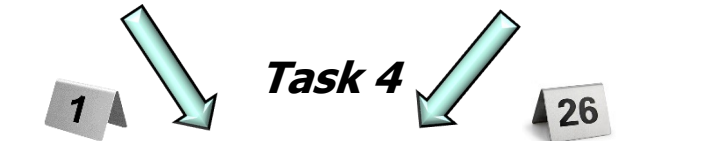


1 **Task 2** /imgNum1\ = /page\ **8**
.thenApplyAsync
(countImages(page))
.thenApply(List::size)

26 **Task 3** /imgNum2\ = /page\ **8**
.thenComposeAsync
(crawlHyperLinks
(page))



1 **Task 4** **26**
/imgNum1\.thenCombine(/imgNum2\
(imgNum1, imgNum2) ->
Integer::sum)



Overview of the Java Completable Futures Framework

- In contrast, Java's completable futures framework supports dependent actions that trigger upon completion of async operations

- These async operations can run in a thread pool
- The time needed to perform these tasks depends on how well tasks can be parallelized



```
8 /page\ = Task 1  
supplyAsync  
(getStartPage())
```

```
1 Task 2  
/imgNum1\ = /page\ 8  
.thenApplyAsync  
(countImages(page))  
.thenApply(List::size)
```

```
26 Task 3  
/imgNum2\ = /page\ 8  
.thenComposeAsync  
(crawlHyperLinks  
(page))
```

```
1 Task 4  
/imgNum1\.thenCombine(/imgNum2\  
(imgNum1, imgNum2) ->  
Integer::sum)
```

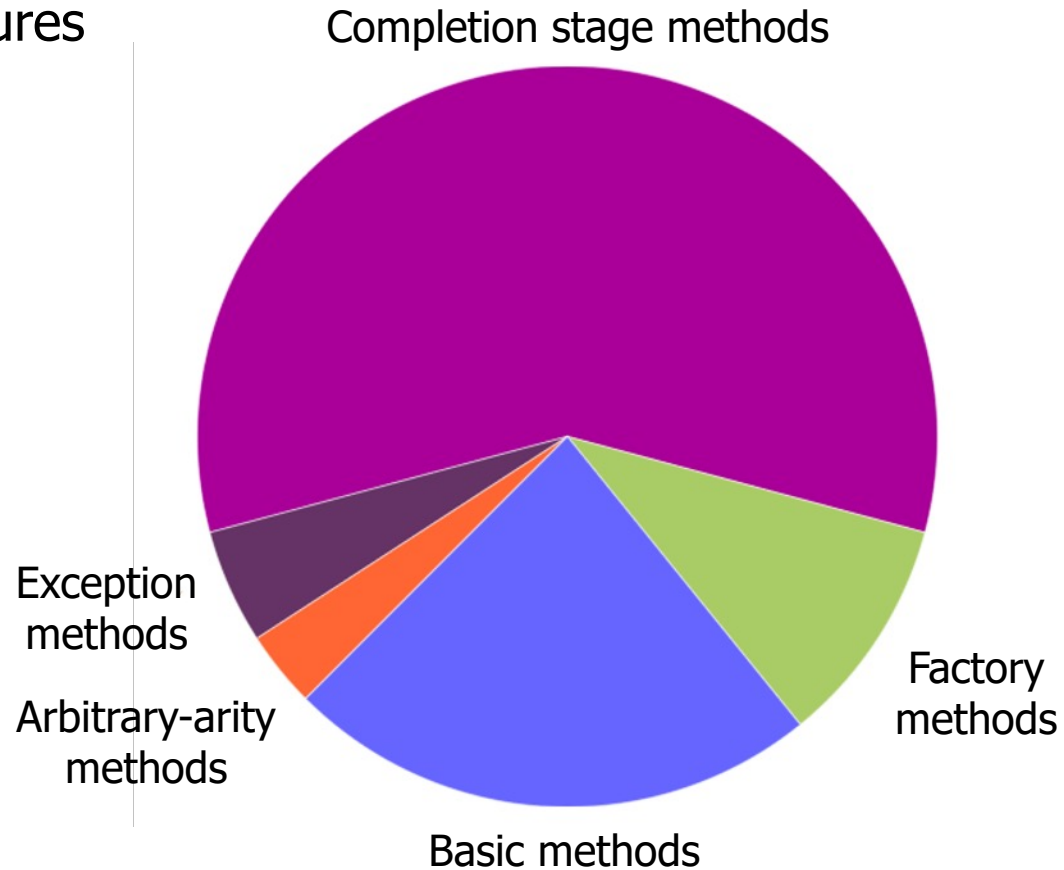
$$\text{Speedup}(N) = \frac{1}{(1-P) + \frac{P}{N}}$$

Serial part of job =
1 (100%) - Parallel part

Parallel part is divided
up by N workers

Overview of the Java Completable Futures Framework

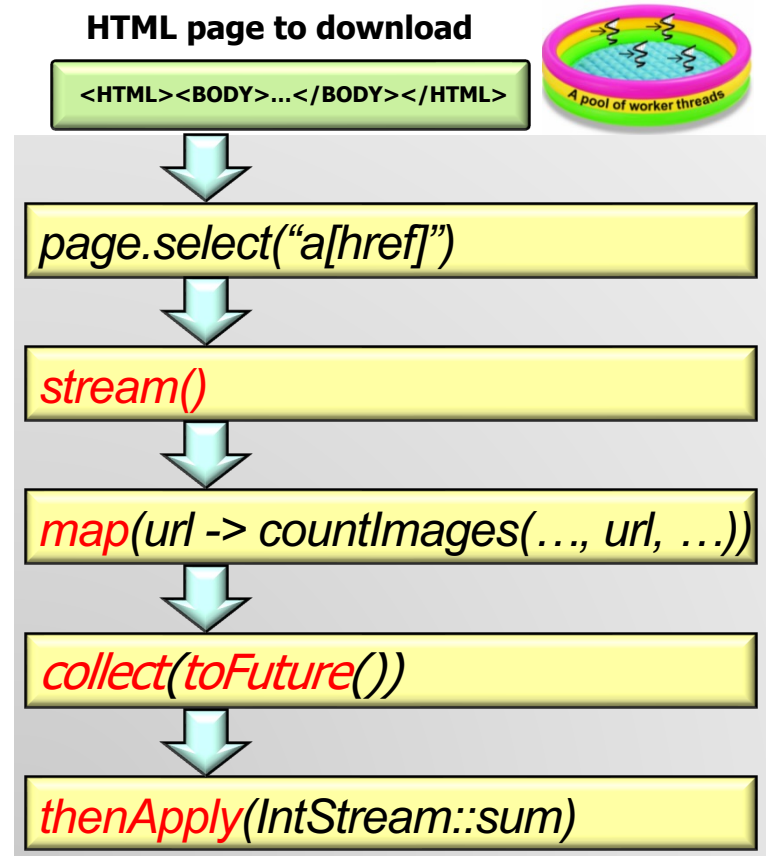
- The entire Java completable futures framework resides in one public class with 60+ methods



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

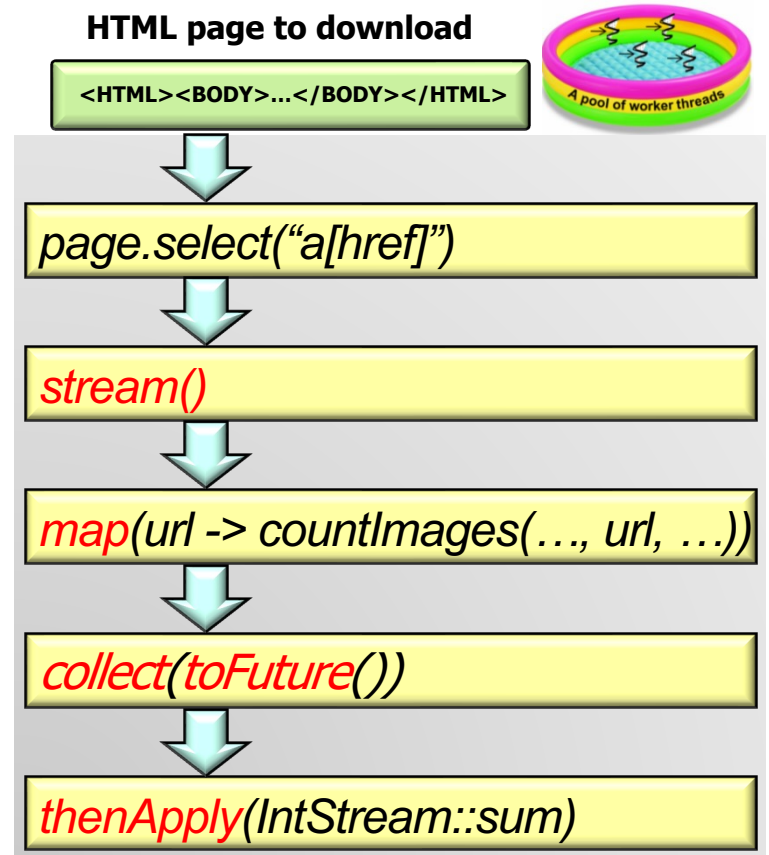
Overview of the Java Completable Futures Framework

- Java completable futures, sequential streams, & functional programming features can be combined nicely!!



Overview of the Java Completable Futures Framework

- Java completable futures often need no explicit synchronization or threading when developing parallel programs!

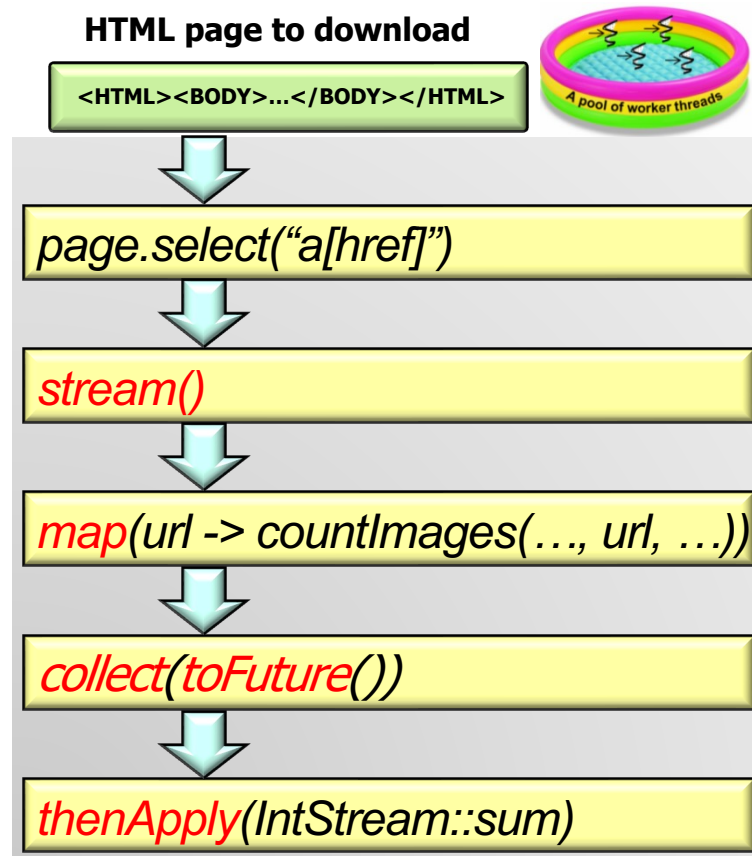



Alleviates many accidental & inherent complexities of parallel programming

Overview of the Java Completable Futures Framework

- Java completable futures often need no explicit synchronization or threading when developing parallel programs!

Java Language									
java	javac	javadoc	apt	jar	javap	JPDA	JConsole		
Security	Int'l	RMI	IDL	Deploy	Monitoring	Troubleshoot	Scripting	JVM TI	
Deployment		Java Web Start				Java Plug-in			
AWT			Swing			Java 2D			
Accessibility	Drag n Drop	Input Methods	Image I/O	Print Service	Sound				
IDL	JDBC	JNDI	RMI	RMI-IIOP					
Beans	Intl Support	Input Output	JMX	JNI	Math				
Networking	Override Mechanism	Security	Serialization	Extension Mechanism	XML JAXP				
lang and util	Collections	Concurrency Utilities	JAR	Logging	Management				
Preferences API	Ref Objects	Reflection	Regular Expressions	Versioning	Zip	Instrumentation			



Java class libraries handle locking needed to protect shared mutable state

End of the Structure & Functionality of the Java Completable Futures Framework