Learning Objectives in this Part of the Lesson

• Understand the pros & cons of using the completable futures framework
Pros & Cons of Java 8
Completable Futures
No explicit synchronization or threading is required in this implementation.
Java libraries handle any locking needed to protect shared mutable state.
Pros & Cons of Java 8 Completable Futures

- We’ll now evaluate the Java 8 completable futures framework compared with the parallel streams framework.

**Parallel Streams**

- filter(not(this::urlCached))
- map(this::downloadImage)
- flatMap(this::applyFilters)
- collect(toList())

**Completable Futures**

- map(this::checkUrlCachedAsync)
- map(this::downloadImageAsync)
- flatMap(this::applyFiltersAsync)
- collect(toFuture())
- thenAccept(this::logResults)

void processStream() {
    List<URL> urls = getInput();

    List<Image> filteredImages = urls
        .parallelStream()
        .filter(not(this::urlCached))
        .map(this::blockingDownload)
        .flatMap(this::applyFilters)
        .collect(toList());

    logResults(filteredImages);
}

void processStream() {
    List<URL> urls = getInput();

    CompletableFuture<Stream<Image>> resultsFuture = urls
        .stream()
        .map(this::checkUrlCachedAsync)
        .map(this::downloadImageAsync)
        .flatMap(this::applyFiltersAsync)
        .collect(toFuture())
        .thenApply(this::logResults)
        .join();

    ...
Pros & Cons of Java 8 Completable Futures

- There's a tradeoff between computing performance & programmer productivity when choosing amongst these frameworks.
Pros & Cons of Java 8 Completable Futures

- There's a tradeoff between computing performance & programmer productivity when choosing amongst these frameworks, e.g.
  - Completable futures are more efficient & scalable, but are harder to program

Printing results for input 1 from fastest to slowest:
- COMPLETABLE_FUTURES_2 executed in 276 msecs
- COMPLETABLE_FUTURES_1 executed in 285 msecs
- PARALLEL_STREAM executed in 383 msecs
- SEQUENTIAL_STREAM executed in 1288 msecs

Printing results for input 2 from fastest to slowest:
- COMPLETABLE_FUTURES_1 executed in 137 msecs
- COMPLETABLE_FUTURES_2 executed in 138 msecs
- PARALLEL_STREAM executed in 170 msecs
- SEQUENTIAL_STREAM executed in 393 msecs
Pros & Cons of Java 8 Completable Futures

- There's a tradeoff between computing performance & programmer productivity when choosing amongst these frameworks, e.g.
  - Completable futures are more efficient & scalable, but are harder to program
  - Parallel streams are often easier to program, but are less efficient & scalable
Pros & Cons of Java 8 Completable Futures

- There's a tradeoff between computing performance & programmer productivity when choosing amongst these frameworks, e.g.
  - Completable futures are more efficient & scalable, but are harder to program
  - Parallel streams are often easier to program, but are less efficient & scalable
  - Combining sequential streams & completable futures is often a win
Pros & Cons of Java 8 Completable Futures

• There's a tradeoff between computing performance & programmer productivity when choosing amongst these frameworks, e.g.
  • Completable futures are more efficient & scalable, but are harder to program
  • Parallel streams are often easier to program, but are less efficient & scalable
  • Combining sequential streams & completable futures is often a win
  • However, it’s overkill to combine parallel streams & completable futures
13

```
CompletableFuture
    .supplyAsync(
        () -> findBestPrice("LDN - NYC"),
        executorService)
    .thenCombine(CompletableFuture
        .supplyAsync
        (() -> queryExchangeRateFor("GBP")),
        this::convert)
    .orTimeout(1, TimeUnit.SECONDS)
    .whenComplete((amount, ex) -> {
        if (ex == null)
            { System.out.println("The price is: " + amount + " GBP"); } 
        else { System.out.println(ex.getMessage()); } 
    });
```

Pros & Cons of Java 8 Completable Futures

- Java 9 fixes some completable future limitations

See iteratrlearning.com/java9/2016/09/13/java9-timeouts-completablefutures.html
End of Pros & Cons of Java 8 Completable Futures