Learning Objectives in this Part of the Lesson

• Recognize key operators defined in—or used with—ParallelFlowables

• Scheduler operators
  • These operators provide the context to run other operators in designated threads & thread pools
    • e.g., Schedulers.io()

These operators also work with the Flowable, ParallelFlowable, Single, & Maybe classes
Key Scheduler Operators for RxJava Reactive Types
Key Scheduler Operators for RxJava Reactive Types

- The Schedulers.io() operator
- Hosts a variable-size pool of single-threaded Executor Service-based workers
Key Scheduler Operators for RxJava Reactive Types

- The Schedulers.io() operator
  - Hosts a variable-size pool of single-threaded Executor Service-based workers
  - Returns a new Scheduler that is suited for I/O-bound work

static Scheduler io()
The Schedulers.io() operator
- Hosts a variable-size pool of single-threaded Executor Service-based workers
- Returns a new Scheduler that is suited for I/O-bound work
- Optimized for blocking operations
Key Scheduler Operators for RxJava Reactive Types

- The Schedulers.io() operator
- Hosts a variable-size pool of single-threaded Executor Service-based workers
- Returns a new Scheduler that is suited for I/O-bound work
- Optimized for blocking operations
- i.e., I/O-bound tasks not compute-/CPU-bound tasks!

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/schedulers/Schedulers.html
Key Scheduler Operators for RxJava Reactive Types

- The Schedulers.io() operator
  - Hosts a variable-size pool of single-threaded Executor Service-based workers
  - Returns a new Scheduler that is suited for I/O-bound work
    - Optimized for blocking operations
    - Either starts a new thread or reuses an idle one from a cache
Key Scheduler Operators for RxJava Reactive Types

- The Schedulers.io() operator
  - Hosts a variable-size pool of single-threaded Executor Service-based workers
  - Returns a new Scheduler that is suited for I/O-bound work
    - Optimized for blocking operations
    - Either starts a new thread or reuses an idle one from a cache
    - The goal is to maximally utilize the CPU cores
Key Scheduler Operators for RxJava Reactive Types

- The Schedulers.io() operator
  - Hosts a variable-size pool of single-threaded Executor Service-based workers
  - Used for making network calls, file I/O, database operations, etc.

```java
return Options.instance()
  .getUrlsFlowable()
  .parallel()
  .runOn(Schedulers.io())
  .map(downloadAndStoreImage)
  .sequential()
  .collect(Collectors.toList())
  .doOnSuccess(...)
```

Download images from remote web servers in parallel & store them on the local computer

See [github.com/dougalcraigschmidt/LiveLessons/tree/master/Reactive/Flowable/ex4](http://github.com/dougalcraigschmidt/LiveLessons/tree/master/Reactive/Flowable/ex4)
The Schedulers.io() operator
- Hosts a variable-size pool of single-threaded Executor Service-based workers
- Used for making network calls, file I/O, database operations, etc.

Create a Flowable containing URLs to download from remote web servers

```
return Options.instance()
    .getUrlFlowable()
    .parallel()
    .runOn(Schedulers.io())
    .map(downloadAndStoreImage)
    .sequential()
    .collect(Collectors.toList())
    .doOnSuccess(...)  
```
Key Scheduler Operators for RxJava Reactive Types

- The Schedulers.io() operator
  - Hosts a variable-size pool of single-threaded Executor Service-based workers
  - Used for making network calls, file I/O, database operations, etc.

```java
return Options.instance().
    getUrlFlowable().
    parallel().
    runOn(Schedulers.io()).
    map(downloadAndStoreImage).
    sequential().
    collect(Collectors.toList()).
    doOnSuccess(...)

Convert the Flowable into a ParallelFlowable
```
Key Schedulers Operators for RxJava Reactive Types

- The Schedulers.io() operator
  - Hosts a variable-size pool of single-threaded Executor Service-based workers
  - Used for making network calls, file I/O, database operations, etc.

```java
return Options.instance() .getUrlFlowable() .parallel() .runOn(Schedulers.io()) .map(downloadAndStoreImage) .sequential() .collect(Collectors.toList()) .doOnSuccess(...) 
```

Designate the I/O Scheduler that will download & store each image in parallel

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/parallel/ParallelFlowable.html#runOn](http://reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/parallel/ParallelFlowable.html#runOn)
Key Scheduler Operators for RxJava Reactive Types

- The Schedulers.io() operator
  - Hosts a variable-size pool of single-threaded Executor Service-based workers
  - Used for making network calls, file I/O, database operations, etc.

```
return Options.instance()
  .getUrlFlowable()
  .parallel()
  .runOn(Schedulers.io())
  .map(downloadAndStoreImage)
  .sequential()
  .collect(Collectors.toList())
  .doOnSuccess(...)
```

Download & store images in parallel
Key Scheduler Operators for RxJava Reactive Types

- The Schedulers.io() operator
  - Hosts a variable-size pool of single-threaded Executor Service-based workers
  - Used for making network calls, file I/O, database operations, etc.

```java
return Options.instance().
  .getUrlFlowable()
  .parallel()
  .runOn(Schedulers.io())
  .map(downloadAndStoreImage)
  .sequential()
  .collect(Collectors.toList())
  .doOnSuccess(...)
```

*Merge the values from each 'rail' in a round-robin fashion & expose it as a regular Flowable sequence*
Key Scheduler Operators for RxJava Reactive Types

- The Schedulers.io() operator
  - Hosts a variable-size pool of single-threaded Executor Service-based workers
  - Used for making network calls, file I/O, database operations, etc.

```
return Options.instance()
    .getUrlsFlowable()
    .parallel()
    .runOn(Schedulers.io())
    .map(downloadAndStoreImage)
    .sequential()
    .collect(Collectors.toList())
    .doOnSuccess(…)
```

Collect the Flowable into a List
Key Scheduler Operators for RxJava Reactive Types

- The Schedulers.io() operator
  - Hosts a variable-size pool of single-threaded Executor Service-based workers
  - Used for making network calls, file I/O, database operations, etc.

```java
return Options.instance()
    .getUrlFlowable()
    .parallel()
    .runOn(Schedulers.io())
    .map(downloadAndStoreImage)
    .sequential()
    .collect(Collectors.toList())
    .doOnSuccess(…)
```

Handle the final 'reduced' results
Key Scheduler Operators for RxJava Reactive Types

- The Schedulers.io() operator
  - Hosts a variable-size pool of single-threaded Executor Service-based workers
  - Used for making network calls, file I/O, database operations, etc.
  - Implemented via “daemon threads”
    - i.e., won’t prevent the app from exiting even if its work isn’t done

See www.baeldung.com/java-daemon-thread
The Schedulers.io() operator
- Hosts a variable-size pool of single-threaded Executor Service-based workers
- Used for making network calls, file I/O, database operations, etc.
- Implemented via “daemon threads”
- The Schedulers.boundedElastic() operator in Project Reactor is similar

```java
public static Scheduler boundedElastic()

The common boundedElastic instance, a Scheduler that dynamically creates a bounded number of ExecutorService-based Workers, reusing them once the Workers have been shut down. The underlying daemon threads can be evicted if idle for more than 60 seconds.

The maximum number of created threads is bounded by a cap (by default ten times the number of available CPU cores, see DEFAULT_BOUNDED_ELASTIC_SIZE). The maximum number of task submissions that can be enqueued and deferred on each of these backing threads is bounded (by default 100K additional tasks, see DEFAULT_BOUNDED_ELASTIC_QUEUESIZE). Past that point, a RejectedExecutionException is thrown.

See projectreactor.io/docs/core/release/api/reactor/core/scheduler/Schedulers.html#boundedElastic
Key Scheduler Operators for RxJava Reactive Types

- **The Schedulers.io() operator**
  - Hosts a variable-size pool of single-threaded Executor Service-based workers
  - Used for making network calls, file I/O, database operations, etc.
  - Implemented via “daemon threads”
  - The Schedulers.boundedElastic() operator in Project Reactor is similar
  - The Java common fork-join pool is also similar

---

```java
public static ForkJoinPool commonPool()

Returns the common pool instance. This pool is statically constructed; its run state is unaffected by attempts to shutdown() or shutdownNow(). However this pool and any ongoing processing are automatically terminated upon program System.exit(int). Any program that relies on asynchronous task processing to complete before program termination should invoke commonPool().awaitQuiescence, before exit.

Returns:
the common pool instance
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#commonPool](http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#commonPool)
The Schedulers.io() operator

- Hosts a variable-size pool of single-threaded Executor Service-based workers
- Used for event-loops, callbacks, & other computational work
- Implemented via “daemon threads”
- The Schedulers.boundedElastic() operator in Project Reactor is similar

- The Java common fork-join pool is also similar
  - When used with the ManagedBlocker mechanism..
End of Key Scheduler Operators for RxJava Reactive Types (Part 3)