Overview of the RxJava AsyncTaskBarrier Framework

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

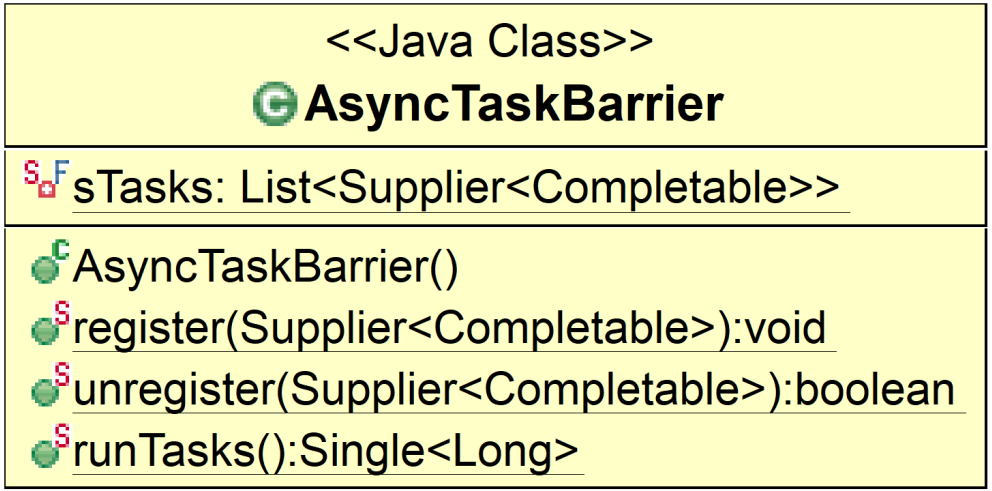
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand key classes in the Project Reactor API
- Understand key classes in the RxJava API
- Be aware of the structure & functionality of the BigFraction case studies
- Recognize the capabilities of the AsyncTaskBarrier framework for RxJava
  - Provides a single location that waits for all asynchronously executing test methods to complete



There are implementations for both Project Reactor & RxJava

Overview of the RxJava AsyncTaskBarrier Class

Overview of the RxJava AsyncTaskBarrier Class

- Most test methods in the BigFraction case studies run asynchronously via subscribeOn(), so these methods return before their computations complete

```
public static Completable testFractionReductionAsync() {
  BigFraction unreducedFraction = makeBigFraction(...);
  ...
  return Single
    .fromCallable(() -> BigFraction.reduce(unreducedFraction))
    .subscribeOn(Schedulers.single())
    .map(result -> result.toMixedString())
    .doOnSuccess(result ->
                 System.out.println
                  ("big fraction = "
                  + result + "\n"))
    .ignoreElement();
```

See Reactive/Single/ex2/src/main/java/SingleEx.java

Overview of the RxJava AsyncTaskBarrier Class

- It's therefore helpful to define a single location in the main driver program that waits for all asynchronously executing test methods to complete

```
public static void main (String[] argv) ... {
  AsyncTaskBarrier
    .register(SingleEx::testFractionReductionAsync);
  AsyncTaskBarrier
    .register(SingleEx::testFractionMultiplicationCallable1);
  AsyncTaskBarrier
    .register(SingleEx::testFractionMultiplicationCallable2);

  long testCount = AsyncTaskBarrier
    .runTasks()
    .blockingGet();
  ...
}
```

See Reactive/Single/ex2/src/main/java/ex2.java

Overview of the RxJava AsyncTaskBarrier Class

- The AsyncTaskBarrier class provides an API to register non-blocking task methods that run *asynchronously*

```
public static void main (String[] argv)
  AsyncTaskBarrier
    .register(SingleEx::testFractionReductionAsync);
  AsyncTaskBarrier
    .register(SingleEx::testFractionMultiplicationCallable1);
  AsyncTaskBarrier
    .register(SingleEx::testFractionMultiplicationCallable2);

  long testCount = AsyncTaskBarrier
    .runTasks()
    .blockingGet();
  ...
}
```

We use all of the register() methods to run async tests

Overview of the RxJava AsyncTaskBarrier Class

- The AsyncTaskBarrier class provides an API to register non-blocking task methods that run *asynchronously*

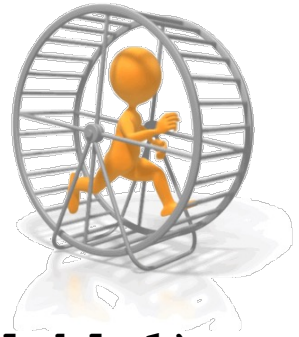
```
public static void main (String[] argv) ... {
  AsyncTaskBarrier
    .register(SingleEx::testFractionReductionAsync);
  AsyncTaskBarrier
    .register(SingleEx::testFractionMultiplicationCallable1);
  AsyncTaskBarrier
    .register(SingleEx::testFractionMultiplicationCallable2);

  long testCount = AsyncTaskBarrier
    .runTasks()
    .blockingGet();
  ...
}
```

This framework also handles task methods that run and/or block *synchronously*

Overview of the RxJava AsyncTaskBarrier Class

- All of the registered task methods start running (a)synchronously when AsyncTaskBarrier.runTasks() is called



```
public static void main (String[] argv) ... {
  AsyncTaskBarrier
    .register(SingleEx::testFractionReductionAsync);
  AsyncTaskBarrier
    .register(SingleEx::testFractionMultiplicationCallable1);
  AsyncTaskBarrier
    .register(SingleEx::testFractionMultiplicationCallable2);

  long testCount = AsyncTaskBarrier
    .runTasks()
    .blockingGet();
  ...
}
```

This call returns a Single

Overview of the RxJava AsyncTaskBarrier Class

- The driver program then calls blockingGet() on the Single returned from runTasks() to wait for all asynchronous task processing to complete

```
public static void main (String[] argv) ... {
  AsyncTaskBarrier
    .register(SingleEx::testFractionReductionAsync);
  AsyncTaskBarrier
    .register(SingleEx::testFractionMultiplicationCallable1);
  AsyncTaskBarrier
    .register(SingleEx::testFractionMultiplicationCallable2);

  long testCount = AsyncTaskBarrier
    .runTasks()
    .blockingGet();
  ...
}
```

It's essential to prevent main() from returning until all the async tasks complete

Overview of the RxJava AsyncTaskBarrier Class

- AsyncTaskBarrier provides a framework that (a)synchronously runs tasks & ensures the calling method doesn't exit until all async processing completes

Class AsyncTaskBarrier

```
public class AsyncTaskBarrier
extends java.lang.Object
```

This class asynchronously runs tasks that use the RxJava framework and ensures that the calling method doesn't exit until all asynchronous task processing is completed.

Method Summary

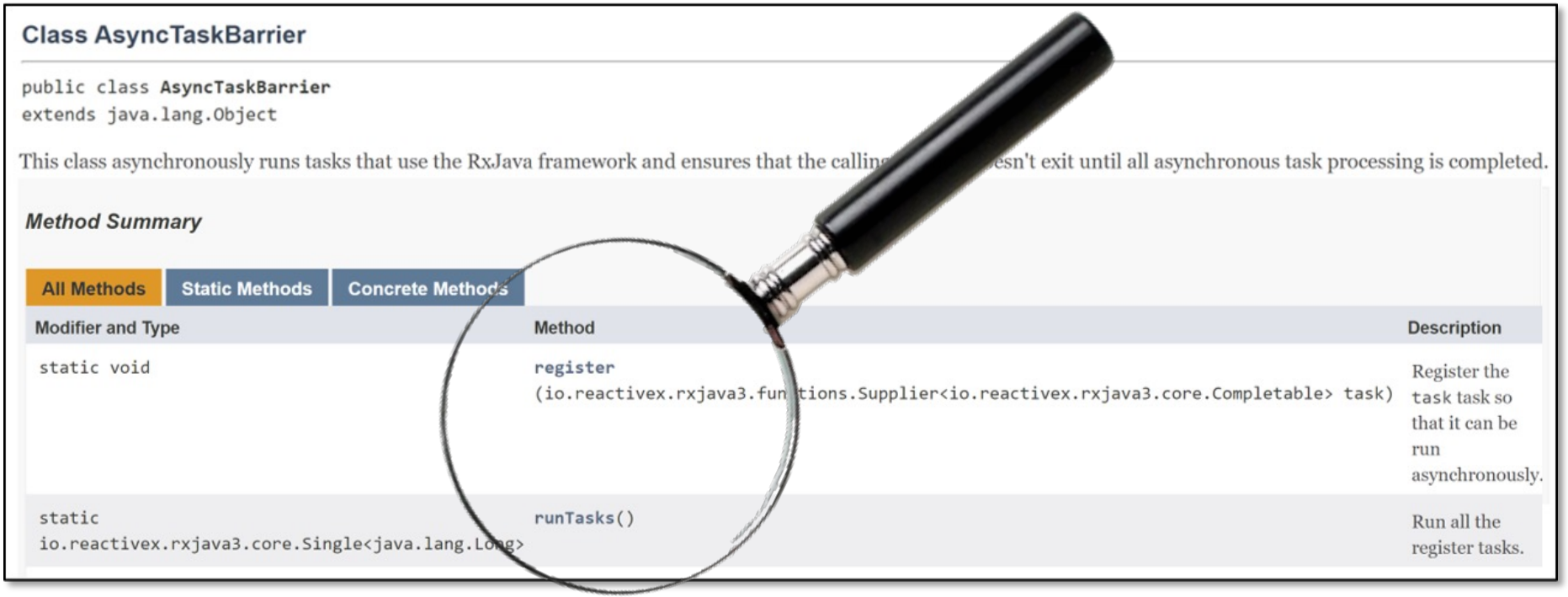
All Methods | Static Methods | Concrete Methods

Modifier and Type	Method	Description
`static void`	`register (io.reactivex.rxjava3.functions.Supplier<io.reactivex.rxjava3.core.Completable> task)`	Register the `task` task so that it can be run asynchronously.
`static io.reactivex.rxjava3.core.Single<java.lang.Long>`	`runTasks()`	Run all the register tasks.

See Reactive/Single/ex2/src/main/java/utils/AsyncTaskBarrier.java

Overview of the RxJava AsyncTaskBarrier Class

- AsyncTaskBarrier provides a framework that (a)synchronously runs tasks & ensures the calling method doesn't exit until all async processing completes



Class AsyncTaskBarrier

```
public class AsyncTaskBarrier
extends java.lang.Object
```

This class asynchronously runs tasks that use the RxJava framework and ensures that the calling [illegible]esn't exit until all asynchronous task processing is completed.

Method Summary

All Methods | Static Methods | Concrete Methods

Modifier and Type	Method	Description
static void	register (io.reactivex.rxjava3.fun[illegible]tions.Supplier<io.reactivex.rxjava3.core.Completable> task)	Register the task task so that it can be run asynchronously.
static io.reactivex.rxjava3.core.Single<java.lang.Long>	runTasks()	Run all the register tasks.

We'll explore AsyncTaskBarrier's implementation after covering RxJava in detail

End of Overview of the RxJava AsyncTaskBarrier Framework