

# Implementing the AsyncTaskBarrier Framework Using RxJava (Part 1)

**Douglas C. Schmidt**

**[d.schmidt@vanderbilt.edu](mailto:d.schmidt@vanderbilt.edu)**

**[www.dre.vanderbilt.edu/~schmidt](http://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 API of the AsyncTaskBarrier class for RxJava

## 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	<b>register</b> (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>	<b>runTasks()</b>	Run all the register tasks.

See [Reactive/Observable/ex4/src/main/java/Utils/AsyncTaskBarrier.java](#)

# Learning Objectives in this Part of the Lesson

---

- Understand the API of the `AsyncTaskBarrier` class for RxJava
- Know how to use `AsyncTaskBarrier` in practice

```
AsyncTaskBarrier.register(this::syncThrowException);  
AsyncTaskBarrier.register(this::asyncThrowException);  
AsyncTaskBarrier.register(this::syncNoException);  
AsyncTaskBarrier.register(this::asyncNoException);
```

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

```
assertEquals(testCount, 2);
```

---

See [Reactive/Observable/ex4/src/test/java/utils/AsyncTaskBarrierTests.java](#)

---

# The AsyncTask Barrier Class API

# The AsyncTaskBarrier Class API

- The AsyncTaskBarrier API contains methods that register, unregister, & (a)synchronously run tasks
- It provides methods to register & unregister tasks







## Interface Supplier<T>

### Type Parameters:

T - the type of results supplied by this supplier







### Functional Interface:

This is a functional interface and can therefore be used as the assignment target for a lambda expression or method reference.

c  AsyncTaskBarrier	
m 	AsyncTaskBarrier()
f 	sTasks List<Supplier<Completable>>
m 	register(Supplier<Completable>) void
m 	runTasks() Single<Long>
m 	unregister(Supplier<Completable>) boolean







# The AsyncTaskBarrier Class API

- The AsyncTaskBarrier API contains methods that register, unregister, & (a)synchronously run tasks

c  AsyncTaskBarrier	
m 	AsyncTaskBarrier()
f 	<i>sTasks</i> List<Supplier<Completable>>
m 	register(Supplier<Completable>) void
m 	runTasks() Single<Long>
m 	unregister(Supplier<Completable>) boolean

# The AsyncTaskBarrier Class API







- The AsyncTaskBarrier API contains methods that register, unregister, & (a)synchronously run tasks
- It provides methods to register & unregister tasks
  - These tasks are stored in a List

c  AsyncTaskBarrier	
m 	AsyncTaskBarrier()
f 	<b>sTasks</b> List<Supplier<Completable>>
m 	register(Supplier<Completable>) void
m 	runTasks() Single<Long>
m 	unregister(Supplier<Completable>) boolean

See [docs.oracle.com/javase/8/docs/api/java/util/List.html](https://docs.oracle.com/javase/8/docs/api/java/util/List.html)

# The AsyncTaskBarrier Class API







- The AsyncTaskBarrier API contains methods that register, unregister, & (a)synchronously run tasks
  - It provides methods to register & unregister tasks
  - It also provides a method that runs all registered tasks (a)synchronously

c  AsyncTaskBarrier	
m 	AsyncTaskBarrier()
f 	sTasks List<Supplier<Completable>>
m 	register(Supplier<Completable>) void
m 	runTasks() Single<Long>
m 	unregister(Supplier<Completable>) boolean



# The AsyncTaskBarrier Class API

- The AsyncTaskBarrier API contains methods that register, unregister, & (a)synchronously run tasks
  - It provides methods to register & unregister tasks
  - It also provides a method that runs all registered tasks (a)synchronously
    - This method doesn't block

c  AsyncTaskBarrier	
m 	AsyncTaskBarrier()
f 	<i>sTasks</i> List<Supplier<Completable>>
m 	register(Supplier<Completable>) void
m 	<b>runTasks()</b> Single<Long>
m 	unregister(Supplier<Completable>) boolean



# The AsyncTaskBarrier Class API

- The AsyncTaskBarrier API contains methods that register, unregister, & (a)synchronously run tasks
  - It provides methods to register & unregister tasks
  - It also provides a method that runs all registered tasks (a)synchronously
    - This method doesn't block
    - When combined with Single.blockingGet() the calling thread won't exit until all asynchronous task processing completes

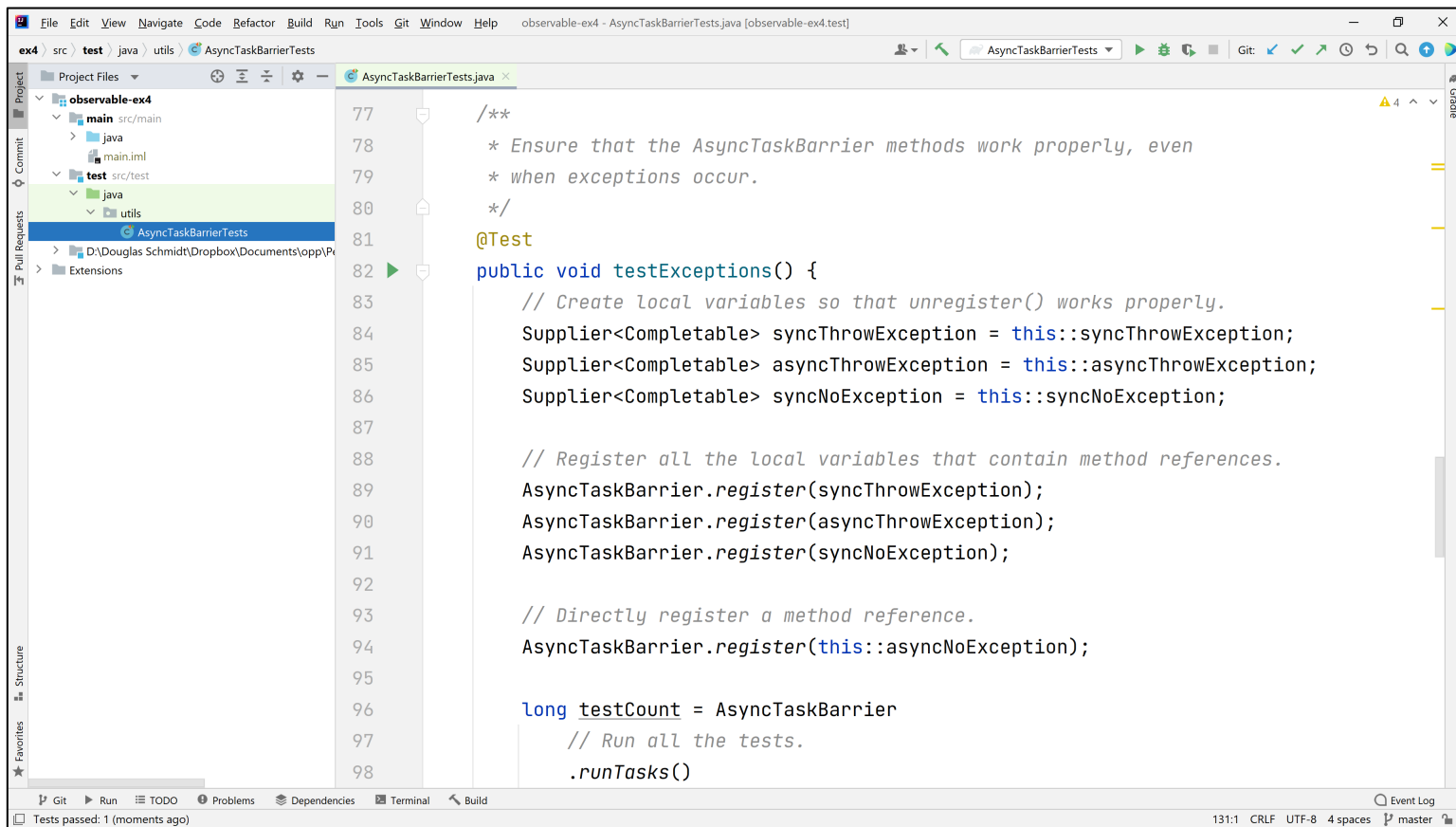
AsyncTaskBarrier	
m	AsyncTaskBarrier()
f	sTasks List<Supplier<Completable>>
m	register(Supplier<Completable>) void
m	<b>runTasks()</b> Single<Long>
m	unregister(Supplier<Completable>) boolean



---

# Applying the `AsyncTask` Barrier in Practice

# Applying the AsyncTaskBarrier in Practice



```
77  /**
78   * Ensure that the AsyncTaskBarrier methods work properly, even
79   * when exceptions occur.
80   */
81  @Test
82  public void testExceptions() {
83      // Create local variables so that unregister() works properly.
84      Supplier<Completable> syncThrowException = this::syncThrowException;
85      Supplier<Completable> asyncThrowException = this::asyncThrowException;
86      Supplier<Completable> syncNoException = this::syncNoException;
87
88      // Register all the local variables that contain method references.
89      AsyncTaskBarrier.register(syncThrowException);
90      AsyncTaskBarrier.register(asyncThrowException);
91      AsyncTaskBarrier.register(syncNoException);
92
93      // Directly register a method reference.
94      AsyncTaskBarrier.register(this::asyncNoException);
95
96      long testCount = AsyncTaskBarrier
97          // Run all the tests.
98          .runTasks()
```

See [Reactive/Observable/ex4/src/test/java/utills/AsyncTaskBarrierTests.java](https://github.com/reactive/reactive-examples/tree/master/observable-ex4/src/test/java/utills/AsyncTaskBarrierTests.java)

---

# End of Implementing the AsyncBarrierTask Framework Using RxJava (Part 1)