Applying Key Operators in Project Reactor: Case Study ex4 (Part 1)
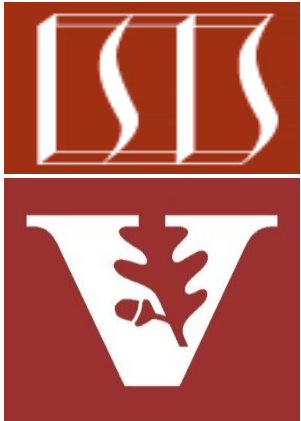
Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

- Part 1 of case study ex4 shows how to implement a generic blocking subscriber

```
class BlockingSubscriber<T>
       implements Subscriber<T> {
  ...
  final CountDownLatch mLatch;
  ...
  @Override
  public void onComplete() {
    ...
    mLatch.countDown();
  }
  ...
  public Mono<Void> await() {
    ...
    mLatch.await();
    ...
```

Learning Objectives in this Part of the Lesson

- Part 1 of case study ex4 shows how to implement a generic blocking subscriber
  - This class is applied to workaround the lack of a blockingSubscribe() operator in Project Reactor Flux
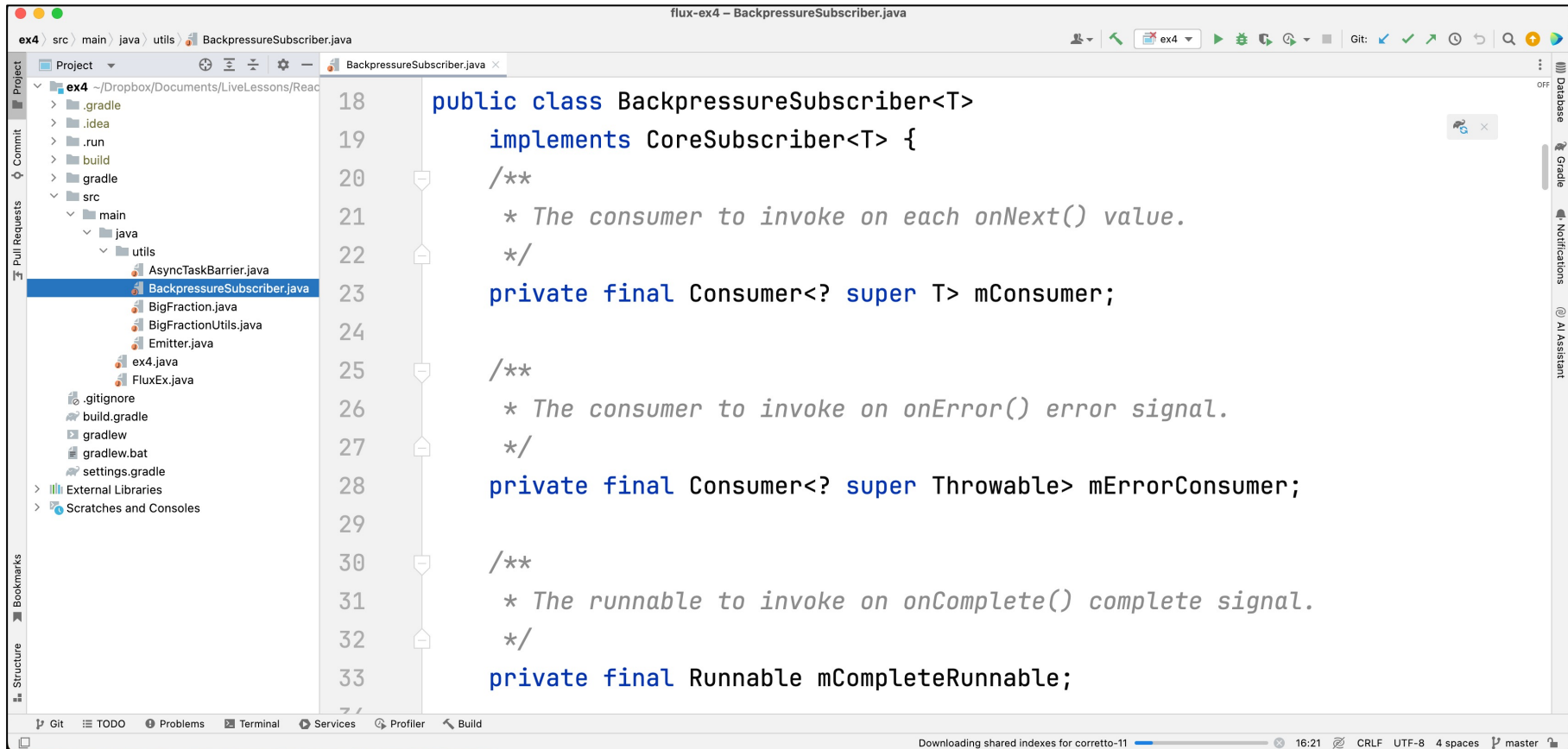
```
return blockingSubscriber
    .await();
```

```
class BlockingSubscriber<T>
        implements Subscriber<T> {
  ...
  final CountDownLatch mLatch;
  ...
  @Override
  public void onComplete() {
    ...
    mLatch.countDown();
  }
  ...
  public Mono<Void> await() {
    ...
    mLatch.await();
    ...
```

In contrast, RxJava has a blockingSubscribe() operator on Observable

Applying Key Operators in Project Reactor to ex4

Applying Key Operators in Project Reactor to ex4

```java
18  public class BackpressureSubscriber<T>
19      implements CoreSubscriber<T> {
20      /**
21       * The consumer to invoke on each onNext() value.
22       */
23      private final Consumer<? super T> mConsumer;
24
25      /**
26       * The consumer to invoke on onError() error signal.
27       */
28      private final Consumer<? super Throwable> mErrorConsumer;
29
30      /**
31       * The runnable to invoke on onComplete() complete signal.
32       */
33      private final Runnable mCompleteRunnable;
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Reactive/flux/ex4

End of Applying Key Methods in Project Reactor: Case Study ex4 (Part 1)