Applying Key Operators in the Flowable Class: Case Study ex1

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

- Case study ex1 applies RxJava Flowable features to demonstrate various types of backpressure strategies (e.g., MISSING, BUFFER, ERROR, LATEST, & DROP) between a Publisher & a Subscriber that run in the context of different Scheduler objects

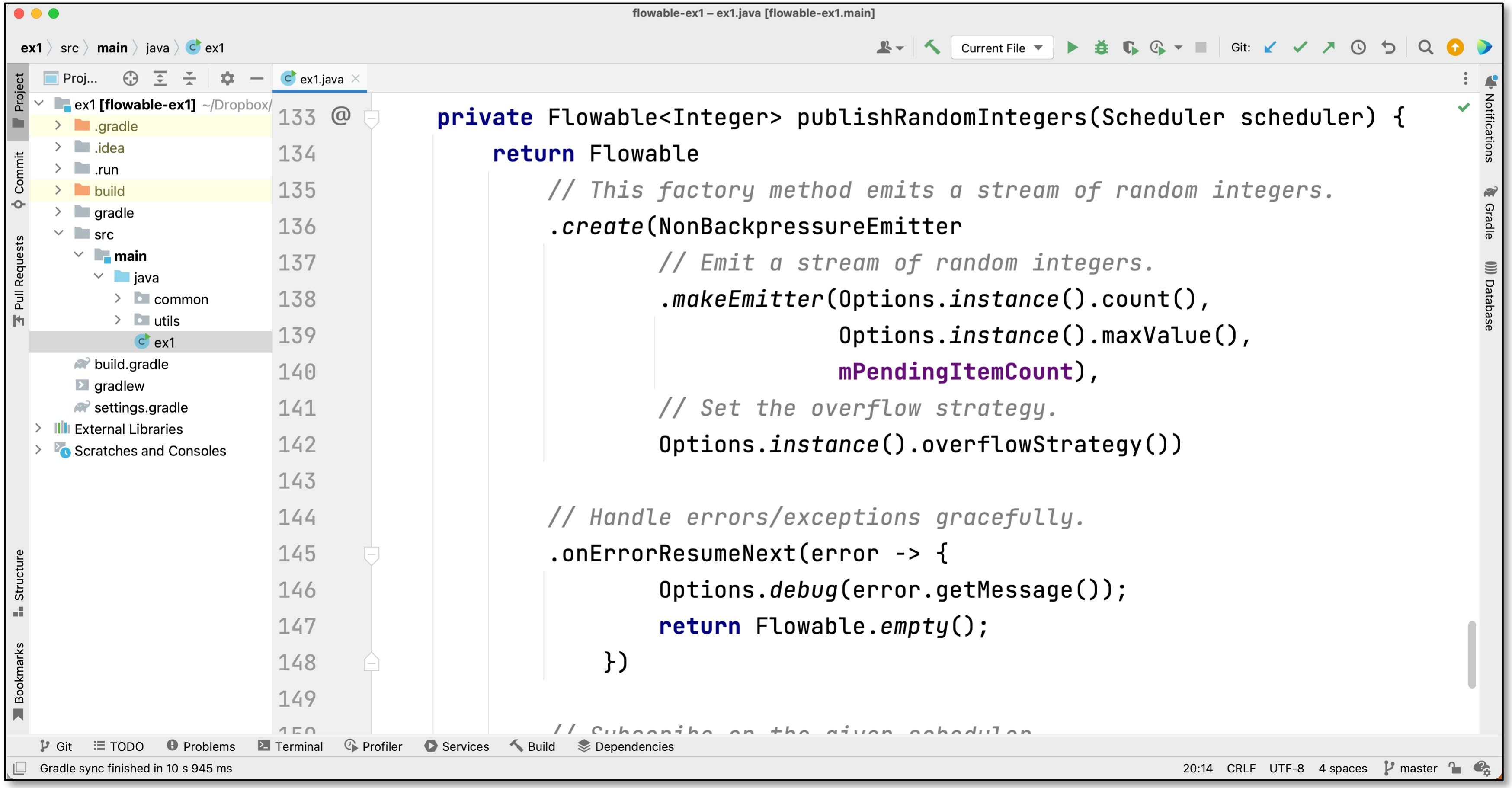
```
return Flowable
  .create(NonBackpressureEmitter
     .makeEmitter(count,
                          maxValue,
                      mPendingItemCount),
      overflowStrategy)

  .onErrorResumeNext(error -> {
     debug(error.getMessage());
     return Flowable.empty();
   })

  .subscribeOn(scheduler);
```

Applying Key Operators in the Flowable Class to ex1

Applying Key Operators in the Flowable Class to ex1



```java
    private Flowable<Integer> publishRandomIntegers(Scheduler scheduler) {
        return Flowable
            // This factory method emits a stream of random integers.
            .create(NonBackpressureEmitter
                    // Emit a stream of random integers.
                    .makeEmitter(Options.instance().count(),
                                 Options.instance().maxValue(),
                                 mPendingItemCount),
                    // Set the overflow strategy.
                    Options.instance().overflowStrategy())

            // Handle errors/exceptions gracefully.
            .onErrorResumeNext(error -> {
                    Options.debug(error.getMessage());
                    return Flowable.empty();
                })

```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Reactive/Flowable/ex1

End of Applying Key Operators in the Flowable Class: Case Study ex1