

# Applying Key Operators in the Flowable Class: Case Study ex1

**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

---

- 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)

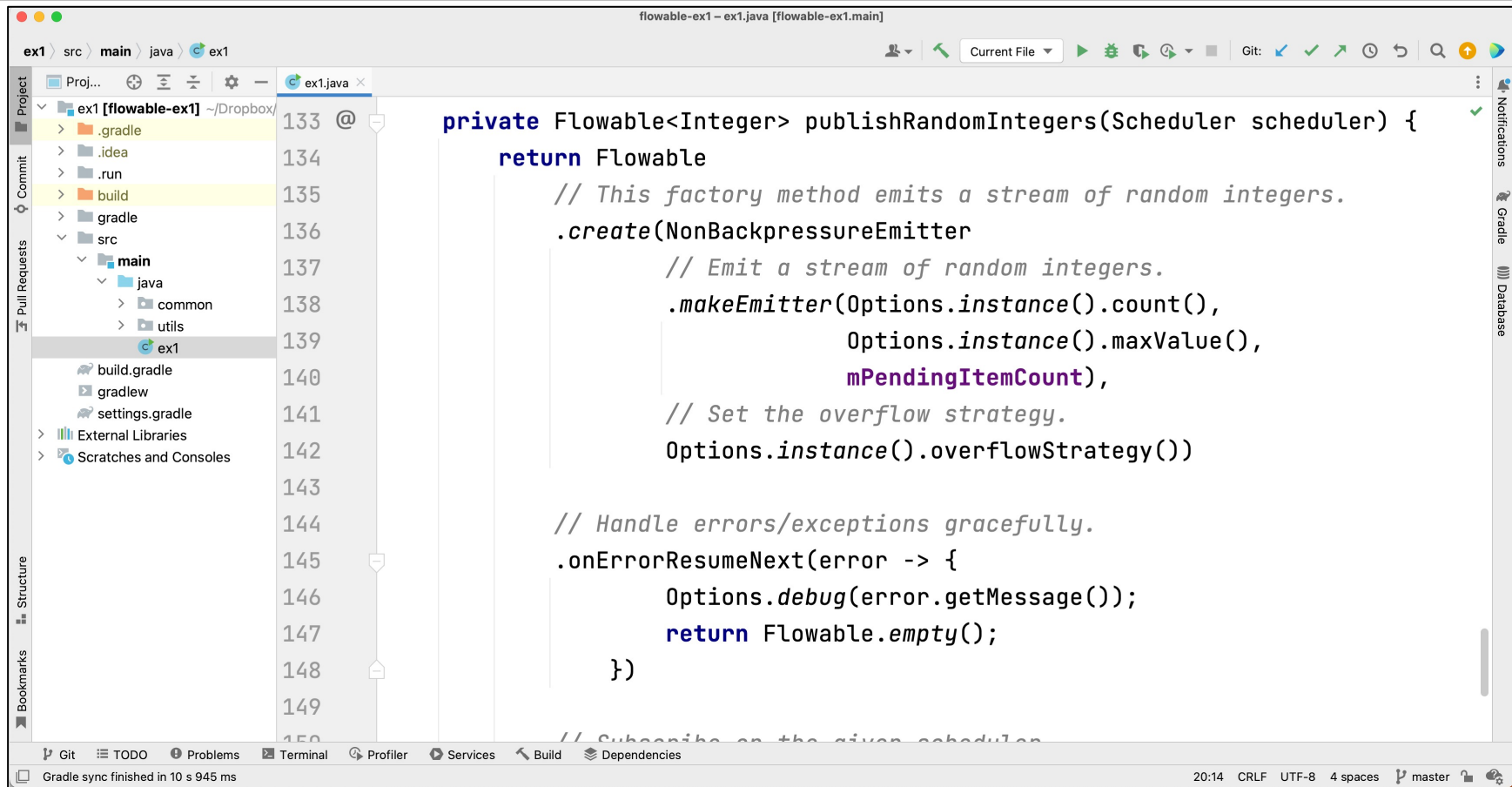
    .onErrorResumeNext (error -> {
        debug (error.getMessage () ) ;
        return Flowable.empty () ;
    })

    .subscribeOn (scheduler) ;
```

---

# Applying Key Operators in the Flowable Class to ex1

# Applying Key Operators in the Flowable Class to ex1



The screenshot shows an IDE window titled "flowable-ex1 - ex1.java [flowable-ex1.main]". The left sidebar displays a project structure for "ex1" with folders like ".gradle", ".idea", ".run", "build", "gradle", "src", "main", "java", "common", and "utils". The "ex1" folder is selected. The main editor area shows the following Java code:

```
133 @
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
```

```
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();
        })
}
```

The bottom status bar indicates "Gradle sync finished in 10 s 945 ms" and "20:14 CRLF UTF-8 4 spaces master".

See [github.com/douglasraigschmidt/LiveLessons/tree/master/Reactive/Flowable/ex1](https://github.com/douglasraigschmidt/LiveLessons/tree/master/Reactive/Flowable/ex1)

---

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