

Applying Key Operators in the Parallel Flux Class: Case Study ex5 (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



Learning Objectives in this Part of the Lesson

- Part 1 of case study ex5 shows how to multiply & add big fractions asynchronously & concurrently using Project Reactor operators, e.g.,
 - Flux operators fromArray(), zipWith(), doOnNext(), doFinally(), then(), & parallel()
 - ParallelFlux operators runOn(), map(), & reduce()
 - The Schedulers.parallel() thread pool

```
return Flux
    .fromArray(bigFractionArray)
    .parallel()
    .runOn
        (Schedulers.parallel())
    .map(bf -> bf
        .multiply(sBigReducedFrac))
    .reduce(BigFraction::add)
    .doOnSuccess(displayResults)
    .then();
```

Applying Key Operators in the ParallelFlux Class to ex5

Applying Key Operators in the ParallelFlux Class to ex5

The screenshot shows the IntelliJ IDEA IDE interface with the following details:

- Project Structure:** The project is named "ex5 [flux-ex5]". The "src/main/java" directory contains several utility classes: BigFractionUtils, DownloadUtils, Options, URL, and Mono. A file named "ParallelFluxEx.java" is currently open.
- Code Editor:** The code for "ParallelFluxEx.java" is displayed, specifically the "testFractionMultiplications" method. It demonstrates the use of the ParallelFlux class by creating an array of BigFraction objects and then displaying their results.
- Toolbars and Status Bar:** The bottom of the screen shows various toolbars (Git, Run, TODO, Problems, Terminal, Services, Profiler, Build, Logcat, Dependencies, App Inspection) and a status bar indicating the current time (79:22), file encoding (CRLF), and branch (master).
- Sidebar:** On the right side, there are several panels: Database, Gradle, Device Manager, Notifications, and Android Emulator.

```
public static Mono<Void> testFractionMultiplications() {
    StringBuffer sb =
        new StringBuffer(str: ">> Calling testFractionMultiplications()\n");

    // Create an array of reduced BigFraction objects.
    BigFraction[] bigFractionArray = {
        BigFraction.valueOf( numerator: 1000, denominator: 30),
        BigFraction.valueOf( numerator: 1000, denominator: 40),
        BigFraction.valueOf( numerator: 1000, denominator: 20),
        BigFraction.valueOf( numerator: 1000, denominator: 10)
    };

    // Display the results.
    Consumer<? super BigFraction> displayResults = result -> {
        sb.append("["
            + Thread.currentThread().getId()
            + "] sum of BigFractions = "
            + result
    );
}
```

See github.com/douglasraigschmidt/LiveLessons/tree/master/Reactive/flux/ex5

End of Applying Key Operators in the ParallelFlux Class: Case Study ex5 (Part 2)