Enhancing Java Completable Futures: Framework Extensibility (Part 2)

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

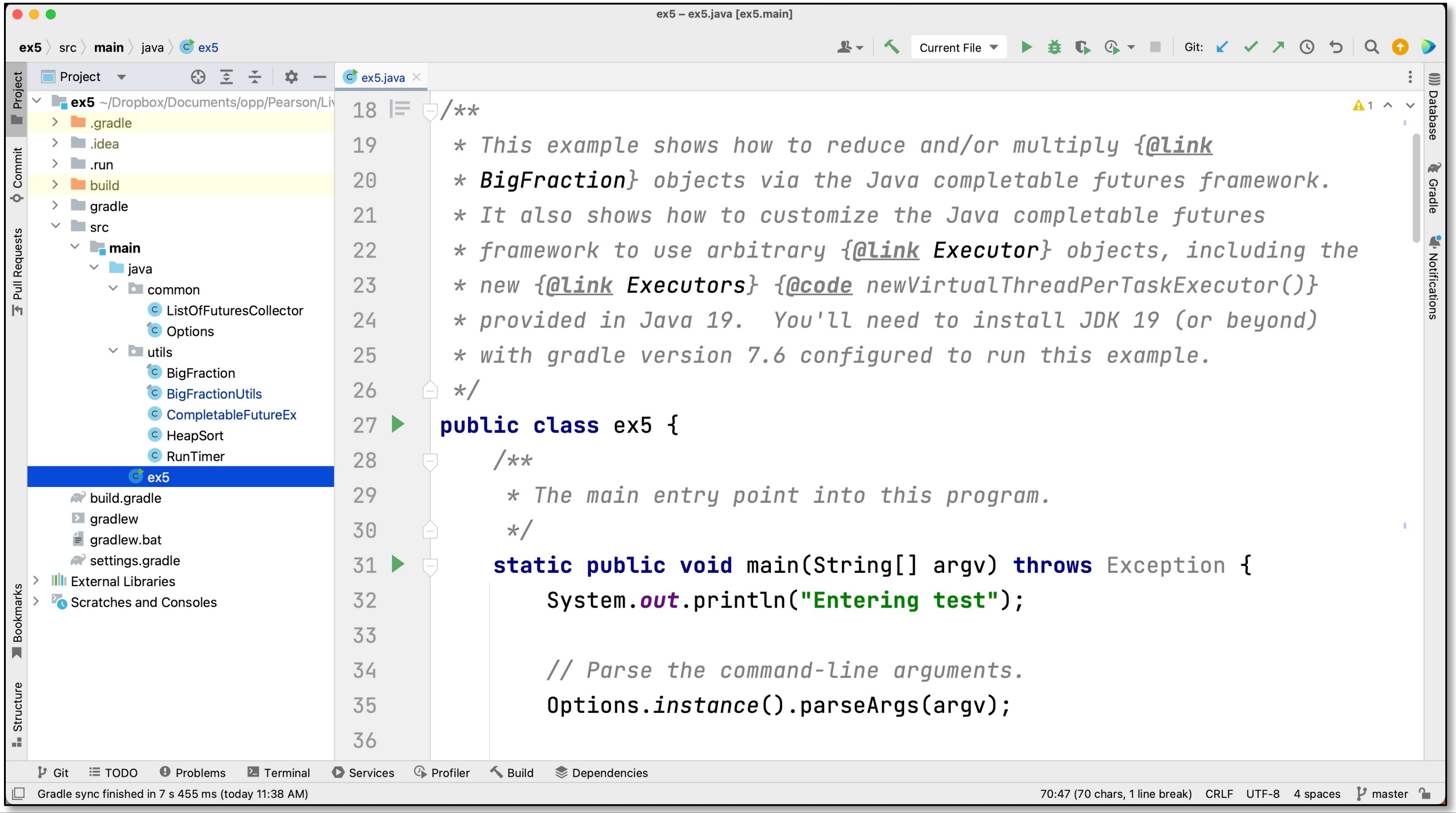
Learning Objectives in this Part of the Lesson

- Evaluate the pros of using the Java completable futures framework
- Evaluate the cons of using the Java completable futures framework
- Understand enhancements to the Java completable futures framework
  - Enhanced timeout handling
  - Enhancing extensibility
    - Analyzing case study ex6 that shows how to apply CompletableFutureEx to reduce & multiply BigFraction objects via Java virtual threads

```
Function<BigFraction,
                  CompletableFuture
                     <BigFraction>>
    reduceAndMultiplyFrac =
      unreducedFrac ->
          CompletableFutureEx
            .supplyAsync(() ->
                BigFraction.reduce
                  (unreducedFrac))
            .thenApplyAsync
               (reducedFrac ->
                reducedFrac
                  .multiply
                  (sBigReducedFrac));
```

Walkthrough of Case Study ex6

Walkthrough of Case Study ex6



```java
/**
 * This example shows how to reduce and/or multiply {@link
 * BigFraction} objects via the Java completable futures framework.
 * It also shows how to customize the Java completable futures
 * framework to use arbitrary {@link Executor} objects, including the
 * new {@link Executors} {@code newVirtualThreadPerTaskExecutor()}
 * provided in Java 19.  You'll need to install JDK 19 (or beyond)
 * with gradle version 7.6 configured to run this example.
 */
public class ex5 {
    /**
     * The main entry point into this program.
     */
    static public void main(String[] argv) throws Exception {
        System.out.println("Entering test");

        // Parse the command-line arguments.
        Options.instance().parseArgs(argv);

```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Loom/ex6

End of Enhancing Java Completable Futures: Framework Extensibility (Part 2)