

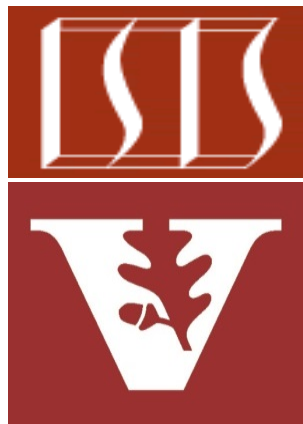
The Java Fork-Join Pool Framework

(Part 3)

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

- Understand how the Java fork-join framework processes tasks in parallel
- Recognize the structure & functionality of the fork-join framework
- Know how the fork-join framework is implemented internally
- Be aware of the common fork-join pool
- Recognize the key methods in the ForkJoinPool class & related classes

```
class ForkJoinPool extends
    AbstractExecutorService {
    ...
    void execute(ForkJoinTask<T>
        task)
    { ... }

    T invoke(ForkJoinTask<T> task)
    { ... }

    ForkJoinTask<T> submit
        (ForkJoinTask<T> task)
    { ... }
```

Learning Objectives in this Part of the Lesson

- Understand how the Java fork-join framework processes tasks in parallel
- Recognize the structure & functionality of the fork-join framework
- Know how the fork-join framework is implemented internally
- Be aware of the common fork-join pool
- Recognize the key methods in the ForkJoinPool class & related classes
- Know how to apply the fork-join framework to perform operations on big fractions

<<Java Class>>

 **BigFraction**

(default package)

 mNumerator: BigInteger

 mDenominator: BigInteger

 valueOf(Number):BigFraction

 valueOf(Number,Number):BigFraction

 valueOf(String):BigFraction

 valueOf(Number,Number,boolean):BigFraction

 reduce(BigFraction):BigFraction

 getNumerator():BigInteger

 getDenominator():BigInteger

 add(Number):BigFraction

 subtract(Number):BigFraction

 multiply(Number):BigFraction

 divide(Number):BigFraction

 gcd(Number):BigFraction

 toMixedString():String

See github.com/douglasraigschmidt/LiveLessons/tree/master/Java8/ex22

Key Methods in Java ForkJoinPool

Key Methods in Java ForkJoinPool

- ForkJoinPool extends AbstractExecutorService

```
class ForkJoinPool extends
    AbstractExecutorService {
    ...
    void execute(Runnable cmd){...}

    <T> Future<T> submit
        (Callable<T> task){...}

    <T> List<Future<T>> invokeAll
        (Collection<? extends
            Callable<T>> tasks){...}

    <T> T invokeAny
        (Collection<? extends
            Callable<T>> tasks){...}
```

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html

Key Methods in Java ForkJoinPool

- ForkJoinPool extends AbstractExecutorService
- It therefore implements the ExecutorService methods

```
class ForkJoinPool extends
    AbstractExecutorService {
    ...
    void execute(Runnable cmd){...}

    <T> Future<T> submit
        (Callable<T> task){...}

    <T> List<Future<T>> invokeAll
        (Collection<? extends
            Callable<T>> tasks){...}

    <T> T invokeAny
        (Collection<? extends
            Callable<T>> tasks){...}
```

Key Methods in Java ForkJoinPool

- ForkJoinPool extends AbstractExecutorService
- It therefore implements the ExecutorService methods



```
class ForkJoinPool extends
    AbstractExecutorService {
    ...
    void execute(Runnable cmd){...}

    <T> Future<T> submit
        (Callable<T> task){...}

    <T> List<Future<T>> invokeAll
        (Collection<? extends
            Callable<T>> tasks){...}

    <T> T invokeAny
        (Collection<? extends
            Callable<T>> tasks){...}
```

However, these methods don't leverage the powerful fork-join pool features

Key Methods in Java ForkJoinPool

- ForkJoinPool extends AbstractExecutorService
 - It therefore implements the ExecutorService methods
 - It also implements key methods for non-ForkJoinTask clients

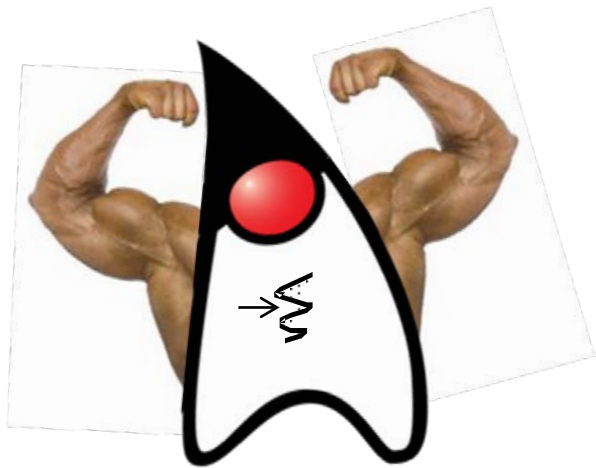
```
class ForkJoinPool extends
    AbstractExecutorService {
    ...
    void execute(ForkJoinTask<T>
                  task)
    { ... }

    T invoke(ForkJoinTask<T> task)
    { ... }

    ForkJoinTask<T> submit
                    (ForkJoinTask<T> task)
    { ... }
```

Key Methods in Java ForkJoinPool

- ForkJoinPool extends AbstractExecutorService
 - It therefore implements the ExecutorService methods
 - It also implements key methods for non-ForkJoinTask clients



```
class ForkJoinPool extends
    AbstractExecutorService {
    ...
    void execute(ForkJoinTask<T>
        task)
    { ... }

    T invoke(ForkJoinTask<T> task)
    { ... }

    ForkJoinTask<T> submit
        (ForkJoinTask<T> task)
    { ... }
```

These methods leverage the powerful properties of the fork-join pool

Key Methods in Java ForkJoinPool

- ForkJoinPool extends AbstractExecutorService
 - It therefore implements the ExecutorService methods
 - It also implements key methods for non-ForkJoinTask clients
 - Arrange async execution

```
class ForkJoinPool extends
    AbstractExecutorService {
    ...
    void execute(ForkJoinTask<T>
                  task)
    { ... }

    T invoke(ForkJoinTask<T> task)
    { ... }

    ForkJoinTask<T> submit
        (ForkJoinTask<T> task)
    { ... }
```

Key Methods in Java ForkJoinPool

- ForkJoinPool extends AbstractExecutorService
 - It therefore implements the ExecutorService methods
 - It also implements key methods for non-ForkJoinTask clients
 - Arrange async execution
 - Performs the task, blocking until it completes

```
class ForkJoinPool extends
    AbstractExecutorService {
    ...
    void execute(ForkJoinTask<T>
                task)
    { ... }

    T invoke(ForkJoinTask<T> task)
    { ... }

    ForkJoinTask<T> submit
        (ForkJoinTask<T> task)
    { ... }
```

Key Methods in Java ForkJoinPool

- ForkJoinPool extends AbstractExecutorService
 - It therefore implements the ExecutorService methods
 - It also implements key methods for non-ForkJoinTask clients
 - Arrange async execution
 - Performs the task, blocking until it completes
 - Submits a ForkJoinTask for execution, returns a future

```
class ForkJoinPool extends
    AbstractExecutorService {
    ...
    void execute(ForkJoinTask<T>
                task)
    { ... }

    T invoke(ForkJoinTask<T> task)
    { ... }

    ForkJoinTask<T> submit
                    (ForkJoinTask<T> task)
    { ... }
```

Key Methods in Java ForkJoinPool

- The ForkJoinPool size defaults to the # of cores available to the JVM

```
class ForkJoinPool extends
    AbstractExecutorService {
    public ForkJoinPool() {
        this(Math.min(MAX_CAP,
            Runtime.getRuntime()
                .availableProcessors()),
            ...);
    }

    public ForkJoinPool
        (int parallelism) {
        this(parallelism, ...);
    }
    ...
}
```



Key Methods in Java ForkJoinPool

- The ForkJoinPool size defaults to the # of cores available to the JVM
- This size can also be controlled programmatically

```
class ForkJoinPool extends
    AbstractExecutorService {
    public ForkJoinPool() {
        this(Math.min(MAX_CAP,
            Runtime.getRuntime()
                .availableProcessors()),
            ...);
    }

    public ForkJoinPool
        (int parallelism) {
        this(parallelism, ...);
    }
    ...
}
```

Key Methods in Java ForkJoinPool

- The common fork-join pool can be accessed via a static method

```
class ForkJoinPool extends
    AbstractExecutorService {
    ...
    static final ForkJoinPool
        common;

    public static ForkJoinPool
        commonPool() {
        return common;
    }
}
```

Key Methods in Java ForkJoinPool

- The common fork-join pool can be accessed via a static method
- The common pool is used by any ForkJoinTask that is not explicitly submitted to a specified pool

```
class ForkJoinPool extends
    AbstractExecutorService {
    ...
    static final ForkJoinPool
        common;

    public static ForkJoinPool
        commonPool() {
        return common;
    }
}
```

Key Methods in Java ForkJoinPool

- ForkJoinPool also provides various management & monitoring operations

int	<u>getParallelism()</u> – Returns the targeted parallelism level of this pool
int	<u>getPoolSize()</u> – Returns the number of worker threads that have started but not yet terminated
int	<u>getQueuedSubmissionCount()</u> – Returns an estimate of the number of tasks submitted to this pool that have not yet begun executing
long	<u>getStealCount()</u> – Returns an estimate of the total number of tasks stolen from one thread's work queue by another

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html

Key Methods in Java ForkJoinTask

Key Methods in Java ForkJoinTask

- ForkJoinTask implements the Future interface

```
abstract class ForkJoinTask<V>  
    implements Future<V>,  
        Serializable {  
    ...  
}
```

Key Methods in Java ForkJoinTask

- ForkJoinTask implements the Future interface
- Asynchronously execute this task in the current task's pool or ForkJoinPool.commonPool()

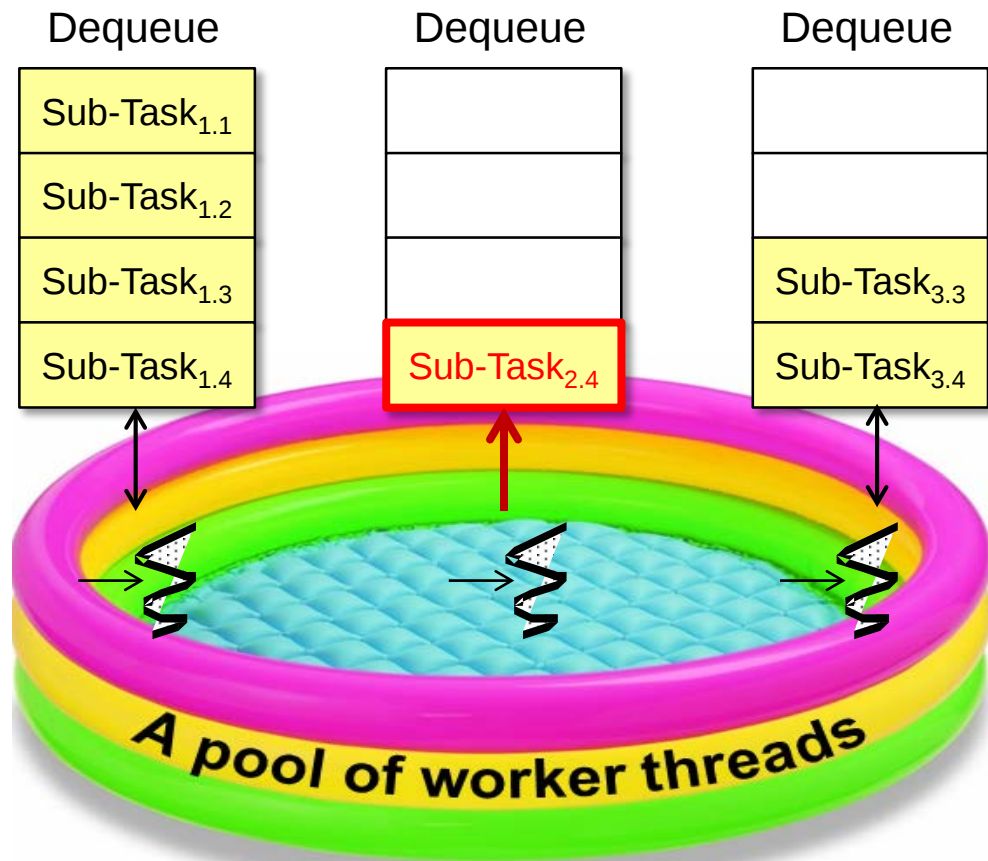
```
abstract class ForkJoinTask<V>
    implements Future<V>,
               Serializable {
    ...
    final ForkJoinTask<V> fork()
    { ... }

    final V join() { ... }

    final V invoke() { ... }
```

Key Methods in Java ForkJoinTask

- ForkJoinTask implements the Future interface
- Asynchronously execute this task in the current task's pool or ForkJoinPool.commonPool()
- fork() pushes the task to the head of the deque in the current thread



Key Methods in Java ForkJoinTask

- ForkJoinTask implements the Future interface
 - Asynchronously execute this task in the current task's pool or ForkJoinPool.commonPool()
- Returns the result of the computation when it's done

```
abstract class ForkJoinTask<V>
    implements Future<V>,
        Serializable {

    ...
    final ForkJoinTask<V> fork()
    { ... }

    final V join() { ... }

    final V invoke() { ... }
```

Key Methods in Java ForkJoinTask

- ForkJoinTask implements the Future interface
 - Asynchronously execute this task in the current task's pool or ForkJoinPool.commonPool()
- Returns the result of the computation when it's done
 - join() causes the current task not to proceed until the forked sub-task has completed

```
abstract class ForkJoinTask<V>
    implements Future<V>,
        Serializable {
    ...
    final ForkJoinTask<V> fork()
    { ... }

    final V join() { ... }

    final V invoke() { ... }
```

Key Methods in Java ForkJoinTask

- ForkJoinTask implements the Future interface
 - Asynchronously execute this task in the current task's pool or ForkJoinPool.commonPool()
 - Returns the result of the computation when it's done
 - Commences performing this task, awaits its completion if necessary, & returns its result

```
abstract class ForkJoinTask<V>
    implements Future<V>,
        Serializable {
    ...
    final ForkJoinTask<V> fork()
    { ... }

    final V join() { ... }

    final V invoke() { ... }
```

Key Methods in Java ForkJoinTask

- ForkJoinTask implements the Future interface
 - Asynchronously execute this task in the current task's pool or ForkJoinPool.commonPool()
 - Returns the result of the computation when it's done
 - Commences performing this task, awaits its completion if necessary, & returns its result
 - Throws RuntimeException or Error if the underlying computation did so

```
abstract class ForkJoinTask<V>
    implements Future<V>,
        Serializable {
    ...
    final ForkJoinTask<V> fork()
    { ... }

    final V join() { ... }

    final V invoke() { ... }
```

Key Methods in the Java RecursiveTask

Key Methods in Java RecursiveTask

- RecursiveTask extends ForkJoinTask to return a result

```
abstract class RecursiveTask<V>  
extends ForkJoinTask<V> {  
...
```

Key Methods in Java RecursiveTask

- RecursiveTask extends ForkJoinTask to return a result
- compute() must be overridden by subclasses to perform the task's main computation

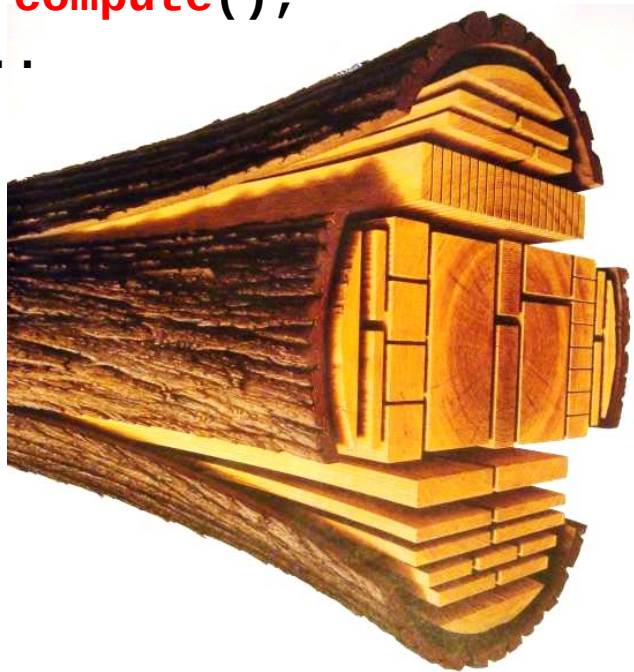
```
abstract class RecursiveTask<V>  
    extends ForkJoinTask<V> {  
    protected abstract V  
        compute();  
    ...  
}
```



Key Methods in Java RecursiveTask

- RecursiveTask extends ForkJoinTask to return a result
- compute() must be overridden by subclasses to perform the task's main computation
- It may split its work up into smaller sub-tasks that are fork()'d to run in parallel

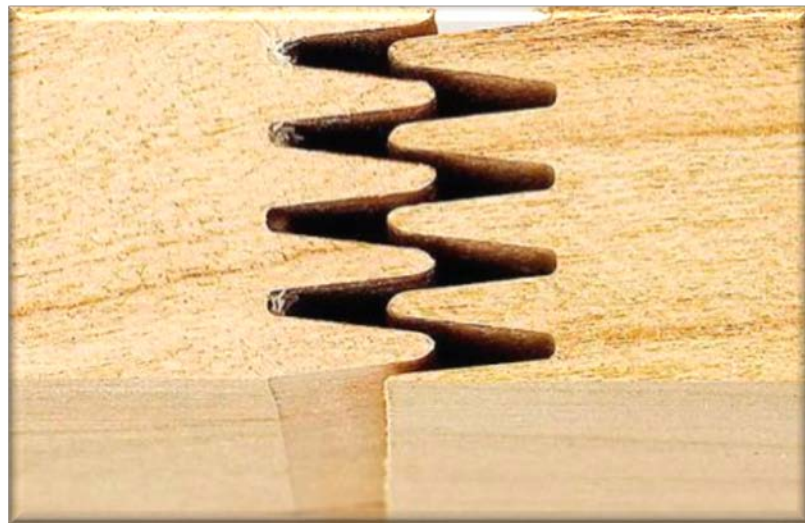
```
abstract class RecursiveTask<V>  
    extends ForkJoinTask<V> {  
    protected abstract V  
        compute();  
    ...  
}
```



Key Methods in Java RecursiveTask

- RecursiveTask extends ForkJoinTask to return a result
- compute() must be overridden by subclasses to perform the task's main computation
 - It may split its work up into smaller sub-tasks that are fork()'d to run in parallel
- It join()'s the results of these smaller sub-tasks into a collective result

```
abstract class RecursiveTask<V>  
    extends ForkJoinTask<V> {  
    protected abstract V  
        compute();  
    ...  
}
```



Key Methods in Java RecursiveTask

- RecursiveTask extends ForkJoinTask to return a result
 - `compute()` must be overridden by subclasses to perform the task's main computation
 - Called internally by the fork-join pool to execute the task

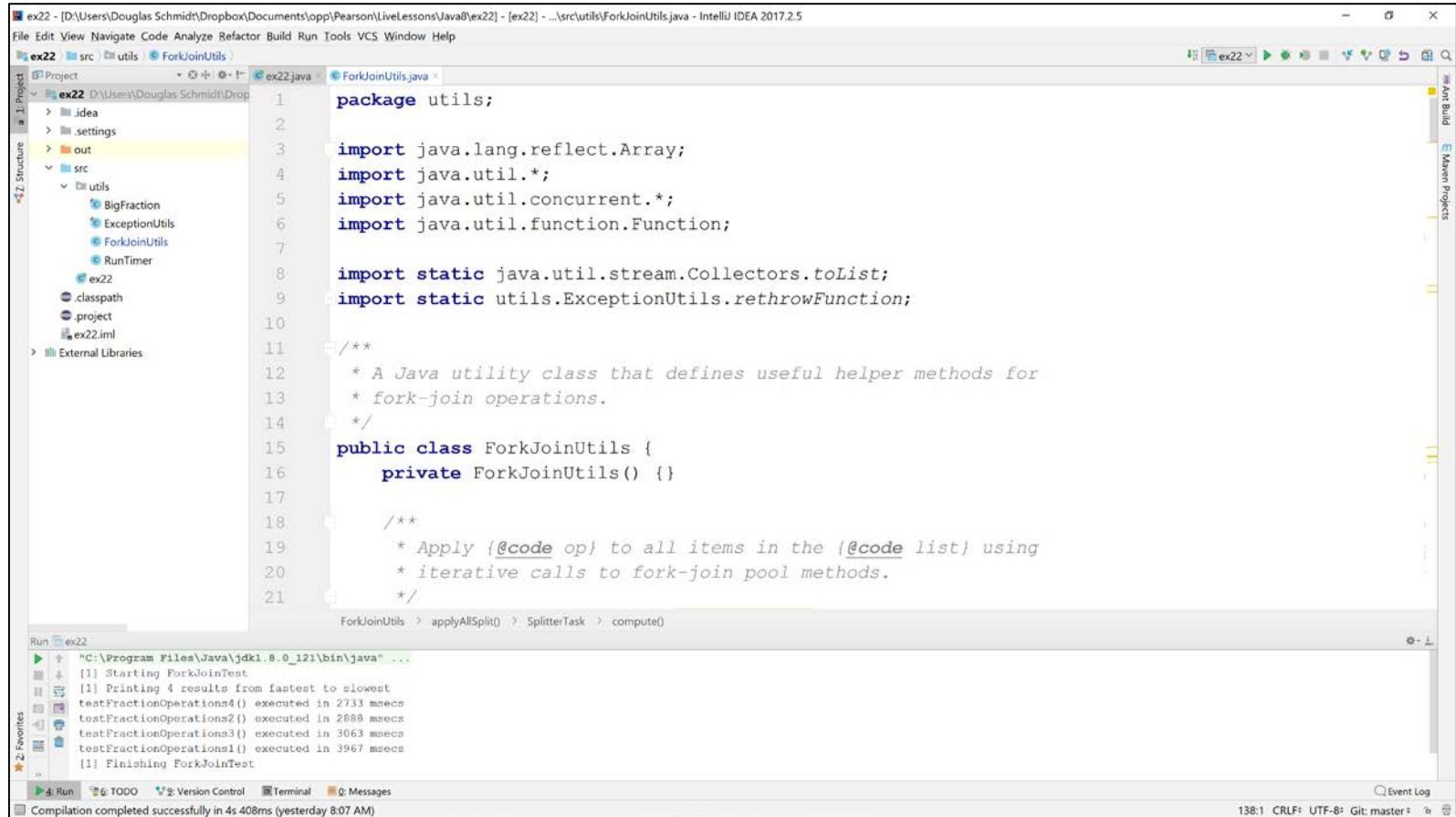
```
abstract class RecursiveTask<V>
    extends ForkJoinTask<V> {
    protected abstract V
        compute();

    V result;

    protected final boolean exec() {
        result = compute();
        return true;
    }
    ...
}
```

Applying the Java Fork-Join Pool

Applying the Java Fork-Join Pool



The screenshot displays the IntelliJ IDEA IDE with the `ForkJoinUtils.java` file open. The code defines a utility class for fork-join operations. The package is `utils`. Imports include `java.lang.reflect.Array`, `java.util.*`, `java.util.concurrent.*`, `java.util.function.Function`, `java.util.stream.Collectors.toList`, and `utils.ExceptionUtils.rethrowFunction`. The class `ForkJoinUtils` has a private constructor. A Javadoc comment describes it as a utility class for fork-join operations. The class is annotated with `@code op` and `@code list` to indicate its use in iterative calls to the fork-join pool methods.

```
1 package utils;
2
3 import java.lang.reflect.Array;
4 import java.util.*;
5 import java.util.concurrent.*;
6 import java.util.function.Function;
7
8 import static java.util.stream.Collectors.toList;
9 import static utils.ExceptionUtils.rethrowFunction;
10
11 /**
12  * A Java utility class that defines useful helper methods for
13  * fork-join operations.
14  */
15 public class ForkJoinUtils {
16     private ForkJoinUtils() {}
17
18     /**
19      * Apply {@code op} to all items in the {@code list} using
20      * iterative calls to fork-join pool methods.
21      */
```

The Run window shows the execution of `ex22`, which starts `ForkJoinTest` and prints 4 results from fastest to slowest. The results are:

- `testFractionOperations4()` executed in 2733 msecs
- `testFractionOperations2()` executed in 2888 msecs
- `testFractionOperations3()` executed in 3063 msecs
- `testFractionOperations1()` executed in 3967 msecs

The Run window also shows the message: `[!] Finishing ForkJoinTest`.

See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex22

End of the Java Fork-Join Pool Framework (Part 3)