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

```java
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

See [github.com/douglasraigschmidt/LiveLessons/tree/master/Java8/ex22](https://github.com/douglasraigschmidt/LiveLessons/tree/master/Java8/ex22)
Key Methods in Java ForkJoinPool
Key Methods in Java ForkJoinPool

- ForkJoinPool extends AbstractExecutorService

```java
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](docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html)
- ForkJoinPool extends AbstractExecutorService
- It therefore implements the ExecutorService methods

```java
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

```java
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

```java
class ForkJoinPool extends AbstractExecutorService {
    ...
    void execute(ForkJoinTask<T> task) {
        ...
    }
    
    T invoke(ForkJoinTask<T> task) {
        ...
    }
    
    ForkJoinTask<T> submit(ForkJoinTask<T> task) {
        ...
    }
}
```
• ForkJoinPool extends AbstractExecutorService
  • It therefore implements the ExecutorService methods
  • It also implements key methods for non-ForkJoinTask clients

```java
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

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

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

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

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#execute](http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#execute)
• 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

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

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

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

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

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

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

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

    ...
}
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#submit](http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#submit)
Key Methods in Java ForkJoinPool

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

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

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

    ...
}
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#ForkJoinPool](docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#ForkJoinPool)
• The ForkJoinPool size defaults to the # of cores available to the JVM
• This size can also be controlled programmatically

```java
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

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

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

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

```java
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

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

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

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

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinTask.html](docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinTask.html)
Key Methods in Java ForkJoinTask

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

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

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinTask.html#fork
```
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 dequeue in the current thread

![Diagram of a pool of worker threads with tasks being dequeued and pushed to the head in a specific order: Sub-Task1.1, Sub-Task1.2, Sub-Task1.3, Sub-Task1.4, Sub-Task2.4, Sub-Task3.3, Sub-Task3.4.]

22
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

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

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinTask.html#join
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
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() {
    ... }

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinTask.html#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
Key Methods in the Java RecursiveTask
Key Methods in Java RecursiveTask

- RecursiveTask extends ForkJoinTask to return a result

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

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html
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();
    ...

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html#compute
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

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

    ...
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html#exec](docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html#exec)
Applying the Java Fork-Join Pool
Applying the Java Fork-Join Pool

```java
package utils;

import java.lang.reflect.Array;
import java.util.*;
import java.util.concurrent.*;
import java.util.function.Function;

import static java.util.stream.Collectors.toList;
import static utils.ExceptionUtils.rethrowFunction;

/**
 * A Java utility class that defines useful helper methods for
 * fork-join operations.
 */
public class ForkJoinUtils {
    private ForkJoinUtils() {} 

    /**
     * Apply [code op] to all items in the [code list] using
     * iterative calls to fork-join pool methods.
     */
```

See [github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex22](https://github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex22)
End of the Java Fork-Join Pool Framework (Part 3)