The Java Fork-Join Pool: Key Methods in ForkJoinPool

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

- Recognize the key methods in the ForkJoinPool class
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)
Key Methods in Java ForkJoinPool

- ForkJoinPool extends Abstract ExecutorService
- 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 Abstract ExecutorService
- It therefore implements the ExecutorService methods
- Arrange async execution of a one-way task

```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){...}
```
ForkJoinPool extends Abstract ExecutorService

It therefore implements the ExecutorService methods

- Arrange async execution of a one-way task
- Submit a two-way task for execution, return a future

```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) {...}
```
ForkJoinPool extends Abstract ExecutorService

It therefore implements the ExecutorService methods

- Arrange async execution of a one-way task
- Submit a two-way task for execution, return a future
- Run all tasks in the collection & wait for them all to finish

```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 Abstract ExecutorService
  - It therefore implements the ExecutorService methods
    - Arrange async execution of a one-way task
    - Submit a two-way task for execution, return a future
    - Run all tasks in the collection & wait for them all to finish
    - Run all tasks in the collection & wait for the first to finish

```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) {...}
```
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 Abstract ExecutorService
  - It therefore implements the ExecutorService methods
- It also implements key methods for non-ForkJoinTask clients

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

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

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

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

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

These methods can leverage the powerful properties of the fork-join pool
ForkJoinPool extends Abstract ExecutorService

- It therefore implements the ExecutorService methods
- It also implements key methods for non-ForkJoinTask clients
- Arrange async execution of one-way task

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

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

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

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#execute
### 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 of one-way task
  - Perform the task, blocking until it completes

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

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

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

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#invoke](https://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 of one-way task
  - Perform the task, blocking until it completes
  - Submit a ForkJoinTask for execution, return a future

```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#submit
Key Methods in Java ForkJoinPool

- The ForkJoinPool size defaults to # of cores available to Java runtime.

```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
Key Methods in Java ForkJoinPool

- The ForkJoinPool size defaults to # of cores available to Java runtime

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

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

    ...
}
```

Returns # of processor cores available to the Java execution environment

See docs.oracle.com/javase/8/docs/api/java/lang/Runtime.html#availableProcessors
The ForkJoinPool size defaults to # of cores available to Java runtime.

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, ...);
    }

    ...
}
```

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#ForkJoinPool
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](docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html#commonPool)
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;
    }
}
```

>This method accesses a static field that can be accessed via all threads in a process}
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.

```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 Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int getParallelism()</td>
<td>- Returns the targeted parallelism level of this pool</td>
</tr>
<tr>
<td>int getPoolSize()</td>
<td>- Returns the number of worker threads that have started but not yet terminated</td>
</tr>
<tr>
<td>int 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 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)
ForkJoinPool also provides various management & monitoring operations

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td><code>getParallelism()</code></td>
<td>Returns the targeted parallelism level of this pool</td>
</tr>
<tr>
<td>int</td>
<td><code>getPoolSize()</code></td>
<td>Returns the number of worker threads that have started but not yet terminated</td>
</tr>
<tr>
<td>int</td>
<td><code>getQueuedSubmissionCount()</code></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><code>getStealCount()</code></td>
<td>Returns an estimate of the total number of tasks stolen from one thread's work queue by another</td>
</tr>
</tbody>
</table>
**Key Methods in Java ForkJoinPool**

- **ForkJoinPool** also provides various management & monitoring operations

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

- **ForkJoinPool** also provides various management & monitoring operations

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

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</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>
End of the Java Fork-Join Pool: Key Methods in ForkJoinPool