The Java ForkJoinPool Class

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
Overview of the ForkJoinPool Class
Overview of the ForkJoinPool Class

- ForkJoinPool implements the ExecutorService interface

**Class ForkJoinPool**

```java
java.lang.Object
   java.util.concurrent.AbstractExecutorService
      java.util.concurrent.ForkJoinPool
```

**All Implemented Interfaces:**
Executor, ExecutorService

```java
public class ForkJoinPool
extends AbstractExecutorService

An ExecutorService for running ForkJoinTasks. A ForkJoinPool provides the entry point for submissions from non-ForkJoinTask clients, as well as management and monitoring operations.

A ForkJoinPool differs from other kinds of ExecutorService mainly by virtue of employing **work-stealing**: all threads in the pool attempt to find and execute tasks submitted to the pool and/or created by other active tasks (eventually blocking waiting for work if none exist). This enables efficient processing when most tasks spawn other subtasks (as do most ForkJoinTasks), as well as when many small tasks are submitted to the pool from external clients. Especially when setting **asyncMode** to true in constructors, ForkJoinPools may also be appropriate for use with event-style tasks that are never joined.

A static **commonPool()** is available and appropriate for most applications. The common pool is used by any ForkJoinTask that is not explicitly submitted to a specified pool. Using the common pool normally reduces resource usage (its threads are slowly reclaimed during periods of non-use, and reinstated upon subsequent use).

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html](http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html)
Overview of the ForkJoinPool Class

• ForkJoinPool implements the ExecutorService interface
• This interface is the basis for Java Executor framework subclasses

See docs.oracle.com/javase/tutorial/essential/concurrency/executors.html
Overview of the ForkJoinPool Class

- ForkJoinPool implements the ExecutorService interface
  - This interface is the basis for Java Executor framework subclasses
- Other implementations of Executor Service execute runnables or callables

<table>
<thead>
<tr>
<th>Java Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runnable</td>
<td>run():void</td>
</tr>
<tr>
<td>Callable&lt;V&gt;</td>
<td>call()</td>
</tr>
<tr>
<td>Executor</td>
<td></td>
</tr>
<tr>
<td>ExecutorService</td>
<td></td>
</tr>
<tr>
<td>AbstractExecutorService</td>
<td></td>
</tr>
<tr>
<td>ThreadPoolExecutor</td>
<td></td>
</tr>
<tr>
<td>ScheduledThreadPoolExecutor</td>
<td></td>
</tr>
<tr>
<td>ForkJoinPool</td>
<td></td>
</tr>
</tbody>
</table>
Overview of the ForkJoinPool Class

- ForkJoinPool implements the ExecutorService interface
  - This interface is the basis for Java Executor framework subclasses
  - Other implementations of Executor Service execute runnables or callables
- In contrast, the ForkJoinPool executes ForkJoinTasks

### Java Interface
- `Executor`

### Java Interface
- `ExecutorService`

### Java Class
- `AbstractExecutorService`

### Java Class
- `ThreadPoolExecutor`

### Java Class
- `ForkJoinPool`

### Java Class
- `ScheduledThreadPoolExecutor`
Overview of the ForkJoinPool Class

- ForkJoinPool implements the ExecutorService interface
  - This interface is the basis for Java Executor framework subclasses
- Other implementations of ExecutorService execute runnables or callables
- In contrast, the ForkJoinPool executes ForkJoinTasks

It can also execute runnables & callables, but that’s not its main purpose
Overview of the ForkJoinPool Class

- There are (intentionally) few “knobs” that can control a ForkJoinPool

<table>
<thead>
<tr>
<th>Java Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ForkJoinPool</td>
</tr>
<tr>
<td>ForkJoinPool()</td>
</tr>
<tr>
<td>ForkJoinPool(int)</td>
</tr>
<tr>
<td>ForkJoinPool(int, ForkJoinWorkerThreadFactory, UncaughtExceptionHandler, boolean)</td>
</tr>
<tr>
<td>commonPool(): ForkJoinPool</td>
</tr>
<tr>
<td>invoke(ForkJoinTask&lt;T&gt;)</td>
</tr>
<tr>
<td>execute(ForkJoinTask&lt;?&gt;): void</td>
</tr>
<tr>
<td>execute(Runnable): void</td>
</tr>
<tr>
<td>submit(ForkJoinTask&lt;T&gt;): ForkJoinTask&lt;T&gt;</td>
</tr>
<tr>
<td>submit(Callable&lt;T&gt;): ForkJoinTask&lt;T&gt;</td>
</tr>
<tr>
<td>submit(Runnable, T): ForkJoinTask&lt;T&gt;</td>
</tr>
<tr>
<td>submit(Runnable): ForkJoinTask&lt;?&gt;</td>
</tr>
<tr>
<td>invokeAll(Collection&lt;Callable&lt;T&gt;&gt;): List&lt;Future&lt;T&gt;&gt;</td>
</tr>
<tr>
<td>shutdown(): void</td>
</tr>
<tr>
<td>shutdownNow(): List&lt;Runnable&gt;</td>
</tr>
<tr>
<td>isTerminated(): boolean</td>
</tr>
<tr>
<td>isTerminating(): boolean</td>
</tr>
<tr>
<td>isShutdown(): boolean</td>
</tr>
<tr>
<td>awaitTermination(long, TimeUnit): boolean</td>
</tr>
</tbody>
</table>
Overview of the ForkJoinPool Class

- There are (intentionally) few “knobs” that can control a ForkJoinPool
- The design goal was to make the ForkJoinPool implementation so clever that programmers can’t improve on its default behavior!
Overview of the ForkJoinPool Class

- In contrast, the ThreadPoolExecutor framework has many control “knobs”
Overview of the ForkJoinPool Class

- In contrast, the ThreadPoolExecutor framework has many control “knobs”

<table>
<thead>
<tr>
<th>&lt;i&gt;Java Class&lt;/i&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThreadPoolExecutor</td>
</tr>
<tr>
<td>ForkJoinPool</td>
</tr>
</tbody>
</table>

- <i>Worker</i>

- <i>Thread Class</i>

See dzone.com/articles/a-deep-dive-into-the-java-executor-service
Overview of the ForkJoinPool Class

- In contrast, the ThreadPoolExecutor framework has many control “knobs”
- The design goal was to enable programmers to maximally customize ThreadPoolExecutor
Overview of the ForkJoinPool Class

- However, you can configure the size of the common fork-join pool
Overview of the ForkJoinPool Class

- However, you *can* configure the size of the common fork-join pool

```java
String desiredThreads = "8";
System.setProperty("java.util.concurrent.ForkJoinPool.common.parallelism",
desiredThreads);
```

Explicitly set the desired # of threads

See lesson on “The Java Fork-Join Pool: Overview of the Common Fork-Join Pool”
Overview of the ForkJoinPool Class

- However, you *can* configure the size of the common fork-join pool

**Interface ForkJoinPool.ManagedBlocker**

Enclosing class:
ForkJoinPool

```java
public static interface ForkJoinPool.ManagedBlocker
```

Interface for extending managed parallelism for tasks running in ForkJoinPools.

*Dynamically adjust the # of threads*

See lesson on “*The Java Fork-Join Pool: the ManagedBlocker Interface*”
End of the Java ForkJoinPool Class