The Java ScheduledExecutor Service (Part 1)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

• Recognize the key methods provided by the Java Scheduled ExecutorService interface & its related interfaces/classes
Learning Objectives in this Part of the Lesson

• Recognize the key methods provided by the Java Scheduled ExecutorService interface & its related interfaces/classes

• Understand the policies defined by the ScheduledThreadPool Executor
Overview of the ScheduledExecutor Service Interface
Overview of the ScheduledExecutorService Interface

- Extends ExecutorService to schedule commands to run after a given delay or to execute periodically

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ScheduledExecutorService.html
Overview of the ScheduledExecutorService Interface

• Key methods in the Scheduled ExecutorService interface

```java
interface ScheduledExecutorService extends ExecutorService {
    public <V> ScheduledFuture<V> schedule(Callable<V> callable,
                                             long delay, TimeUnit unit);

    public ScheduledFuture<?> schedule(Runnable command,
                                         long delay, TimeUnit unit);

    public ScheduledFuture<?> scheduleAtFixedRate(Runnable command,
                                                 long initialDelay,
                                                 long period, TimeUnit unit);

    public ScheduledFuture<?> scheduleWithFixedDelay(Runnable command,
                                                      long initialDelay,
                                                      long delay, TimeUnit unit);
}
```

All schedule() methods accept *relative* delays & periods as arguments, not absolute times or dates.
Overview of the ScheduledExecutorService Interface

- Key methods in the ScheduledExecutorService interface
- Create tasks with various delays

```java
interface ScheduledExecutorService extends ExecutorService {
    public <V> ScheduledFuture<V> schedule(Callable<V> callable,
                                           long delay, TimeUnit unit);

    public <T> ScheduledFuture<T> schedule(Runnable command,
                                           long delay, TimeUnit unit);

    ...
}
```

Tasks can be two-way (callable) or one-way (runnable)
Overview of the ScheduledExecutorService Interface

- Key methods in the ScheduledExecutorService interface
  - Create tasks with various delays
  - Returns an object that can cancel or check execution status

interface ScheduledExecutorService
   extends ExecutorService {
   public <V> ScheduledFuture<V> schedule(Callable<V> callable, 
   long delay, TimeUnit unit);
   public ScheduledFuture<?> schedule(Runnable command, 
   long delay, TimeUnit unit);
   ...

   get() always returns null on a ScheduledFuture returned for a runnable!

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ScheduledFuture.html
Overview of the ScheduledExecutorService Interface

- Key methods in the ScheduledExecutorService interface
  - Create tasks with various delays
  - Returns an object that can cancel or check execution status
  - Some clever design techniques may be needed to cancel the runnable command..

```java
class RefCountedFutureTask<V> extends FutureTask<V> {
    ... 
    /**
     * A scheduled future that can be used to cancel a runnable that has been scheduled to run at a fixed interval to check if the future task has become stale and should be removed from the cache.
     */
    ScheduledFuture<?> mFuture;
```
Overview of the ScheduledExecutorService Interface

- Key methods in the ScheduledExecutorService interface
  - Create tasks with various delays
  - Create & execute tasks that run periodically until cancelled

```java
interface ScheduledExecutorService extends ExecutorService {
    ...

    public ScheduledFuture<?> scheduleAtFixedRate(Runnable command, long initialDelay, long period, TimeUnit unit);

    public ScheduledFuture<?> scheduleWithFixedDelay(Runnable command, long initialDelay, long delay, TimeUnit unit);

    ...
```
Overview of the ScheduledExecutorService Interface

- Key methods in the ScheduledExecutorService interface
  - Create tasks with various delays
  - Create & execute tasks that run periodically until cancelled
  - Next execution starts immediately if the last one takes longer than period

```java
interface ScheduledExecutorService extends ExecutorService {

    public ScheduledFuture<?> scheduleAtFixedRate(Runnable command, long initialDelay, long period, TimeUnit unit);

    public ScheduledFuture<?> scheduleWithFixedDelay(Runnable command, long initialDelay, long delay, TimeUnit unit);

    ...

    ...
}
```
Overview of the ScheduledExecutorService Interface

- Key methods in the ScheduledExecutorService interface
  - Create tasks with various delays
  - Create & execute tasks that run periodically until cancelled
  - Next execution starts immediately if the last one takes longer than period
  - Next execution starts after delay time between termination of one execution & commencement of next

interface ScheduledExecutorService
    extends ExecutorService {
        ...
        public ScheduledFuture<?> scheduleAtFixedRate(Runnable command, long initialDelay, long period, TimeUnit unit);
        
        public ScheduledFuture<?> scheduleWithFixedDelay(Runnable command, long initialDelay, long delay, TimeUnit unit);
        
        ...
    }
Overview of the ScheduledExecutorService Interface

• Key methods in the ScheduledExecutorService interface

• Create tasks with various delays

• Create & execute tasks that run periodically until cancelled
  • Next execution starts immediately if the last one takes longer than period
  • Next execution starts after delay time between termination of one execution & commencement of next

• Again, these methods return an object that can be used to cancel or check the execution status

```java
interface ScheduledExecutorService
    extends ExecutorService {

    public ScheduledFuture<?>
        scheduleAtFixedRate(Runnable command, long initialDelay, long period, TimeUnit unit);

    public ScheduledFuture<?>
        scheduleWithFixedDelay(Runnable command, long initialDelay, long delay, TimeUnit unit);

    ...

    ...
```
Overview of the ScheduledThreadPoolExecutor
Overview of the ScheduledThreadPoolExecutor

- ScheduleThreadPoolExecutor implements ScheduledExecutorService & enables the scheduling of commands that run after a given delay and/or executes them periodically in one or more threads.

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ScheduledThreadPoolExecutor.html
Overview of the ScheduledThreadPoolExecutor

- ScheduledThreadPoolExecutor implements ScheduledExecutorService & enables the scheduling of commands that run after a given delay and/or executes them periodically in one or more threads.

```
ScheduledThreadPoolExecutor
  now():long
  delayedExecute(RunnableScheduledFuture<?>):void
  decorateTask(Callable<V>,RunnableScheduledFuture<V>)
  ScheduledThreadPoolExecutor(int)
  ScheduledThreadPoolExecutor(int,ThreadFactory)
  triggerTime(long,TimeUnit):long
  triggerTime(long):long
  schedule(Runnable,long,TimeUnit)
  schedule(Callable<V>,long,TimeUnit)
  scheduleAtFixedRate(Runnable,long,long,TimeUnit)
  scheduleWithFixedDelay(Runnable,long,long,TimeUnit)
  execute(Runnable):void
  submit(Runnable):Future<?>
  submit(Runnable,T):Future<T>
  submit(Callable<T>):Future<T>
  shutdown():void
  shutdownNow():List<Runnable>
```

```
ScheduledExecutorService
```

```
ThreadPoolExecutor
```

```
ScheduledFutureTask<V>
getDelay(TimeUnit):long
compareTo(Delayed):int
isPeriodic():boolean
setNextRunTime():void
cancel(boolean):boolean
run():void
```

```
DelayedWorkQueue
```

ScheduleThreadPoolExecutor is useful when multiple worker threads are needed or when additional flexibility or policies is needed.
Overview of the ScheduledThreadPoolExecutor

- Apps may require direct access to ScheduledThreadPoolExecutor when its additional flexibility or policies are needed.
Overview of the ScheduledThreadPoolExecutor

- Apps may require direct access to ScheduleThreadPoolExecutor when its additional flexibility or policies are needed, e.g.
- Sets the policy on whether to continue running existing periodic tasks even when executor has been shutdown
Overview of the ScheduledThreadPoolExecutor

• Apps may require direct access to ScheduledThreadPoolExecutor when its additional flexibility or policies are needed, e.g.
  • Sets the policy on whether to continue running existing periodic tasks even when executor has been shutdown
  • Sets the policy on whether to run existing delayed tasks even when executor has been shutdown

### ScheduledThreadPoolExecutor

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>newScheduledThreadPool(int)</td>
<td>Creates a new scheduled thread pool with the specified number of threads.</td>
</tr>
<tr>
<td>newScheduledThreadPool(int, ThreadFactory)</td>
<td>Creates a new scheduled thread pool with the specified number of threads and a custom thread factory.</td>
</tr>
<tr>
<td>newScheduledThreadPool(int, RejectedExecutionPolicy)</td>
<td>Creates a new scheduled thread pool with the specified number of threads and a custom rejection policy.</td>
</tr>
<tr>
<td>newScheduledThreadPool(int, ThreadFactory, RejectedExecutionPolicy)</td>
<td>Creates a new scheduled thread pool with the specified number of threads, a custom thread factory, and a custom rejection policy.</td>
</tr>
<tr>
<td>schedule(Runnable, long, TimeUnit)</td>
<td>Schedules the task for execution after the specified delay.</td>
</tr>
<tr>
<td>schedule(Callable&lt;V&gt;, long, TimeUnit)</td>
<td>Schedules the task for execution after the specified delay and returns a future for the task.</td>
</tr>
<tr>
<td>scheduleAtFixedRate(Runnable, long, long, TimeUnit)</td>
<td>Schedules the task to be executed periodically.</td>
</tr>
<tr>
<td>scheduleWithFixedDelay(Runnable, long, long, TimeUnit)</td>
<td>Schedules the task to be executed with a fixed delay and then periodically.</td>
</tr>
<tr>
<td>execute(Runnable)</td>
<td>Schedules the task for execution.</td>
</tr>
<tr>
<td>submit(Runnable)</td>
<td>Submits the task for execution.</td>
</tr>
<tr>
<td>submit(Runnable, T)</td>
<td>Submits the task for execution with a result type.</td>
</tr>
<tr>
<td>submit(Callable&lt;T&gt;)</td>
<td>Submits the task for execution with a result type and returns a future for the task.</td>
</tr>
</tbody>
</table>

- setContinueExistingPeriodicTasksAfterShutdownPolicy(boolean): void
- getContinueExistingPeriodicTasksAfterShutdownPolicy(): boolean
- setExecuteExistingDelayedTasksAfterShutdownPolicy(boolean): void
- getExecuteExistingDelayedTasksAfterShutdownPolicy(): boolean
- setRemoveOnCancelPolicy(boolean): void
- getRemoveOnCancelPolicy(): boolean
- shutdown(): void
- shutdownNow(): List<Runnable>
- getQueue(): BlockingQueue<Runnable>
Overview of the ScheduledThreadPoolExecutor

- Apps may require direct access to ScheduledThreadPoolExecutor when its additional flexibility or policies are needed, e.g.
  - Sets the policy on whether to continue running existing periodic tasks even when executor has been shutdown
  - Sets the policy on whether to run existing delayed tasks even when executor has been shutdown
  - Sets the policy on whether cancelled tasks should be immediately removed from work queue when cancelled
Overview of the ScheduledThreadPoolExecutor

- Apps may require direct access to ScheduledThreadPoolExecutor when its additional flexibility or policies are needed, e.g.
  - Sets the policy on whether to continue running existing periodic tasks even when executor has been shutdown
  - Sets the policy on whether to run existing delayed tasks even when executor has been shutdown
  - Sets the policy on whether cancelled tasks should be immediately removed from work queue when cancelled

Oddly, all three of these policies are set to false by default.
Overview of the ScheduledThreadPoolExecutor

• Do not try to access a ScheduledThreadPoolExecutor by casting the result of Executors.newSingleThreadScheduledExecutor()

```
ScheduledThreadPoolExecutor schedTPExec =
(ScheduledThreadPoolExecutor)
Executors.newSingleThreadScheduledExecutor();
```

since the implementation of newSingleThreadScheduledExecutor() doesn’t actually return a ScheduledThreadPoolExecutor!

```java
class Executors {
    ...

    public static ScheduledExecutorService
    newSingleThreadScheduledExecutor() {
        return new DelegatedScheduledExecutorService
        (new ScheduledThreadPoolExecutor(1));
    }
}
```

See stackoverflow.com/questions/10200230/creating-scheduledthreadpoolexecutor-using-executors
End of the Java Scheduled ExecutorService (Part 1)