

Overview of Java Scheduled ExecutorService Policies

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 provided by the Java `ScheduledExecutorService` interface & its related interfaces/classes
- Understand the policies defined by the `ScheduledThreadPoolExecutor`

Class `ScheduledThreadPoolExecutor`

```
java.lang.Object
  java.util.concurrent.AbstractExecutorService
    java.util.concurrent.ThreadPoolExecutor
      java.util.concurrent.ScheduledThreadPoolExecutor
```

All Implemented Interfaces:

`Executor`, `ExecutorService`, `ScheduledExecutorService`

```
public class ScheduledThreadPoolExecutor
  extends ThreadPoolExecutor
  implements ScheduledExecutorService
```

A `ThreadPoolExecutor` that can additionally schedule commands to run after a given delay, or to execute periodically. This class is preferable to `Timer` when multiple worker threads are needed, or when the additional flexibility or capabilities of `ThreadPoolExecutor` (which this class extends) are required.

Delayed tasks execute no sooner than they are enabled, but without any real-time guarantees about when, after they are enabled, they will commence. Tasks scheduled for exactly the same execution time are enabled in first-in-first-out (FIFO) order of submission.

Overview of ScheduledThread PoolExecutor & Its Policies

Overview of ScheduledThreadPoolExecutor & Its Policies

- ScheduledExecutorService must be implemented before it is useful (like any interface)

<<Java Interface>>





















ScheduledExecutorService

- `schedule(Runnable,long,TimeUnit)`
- `schedule(Callable<V>,long,TimeUnit)`
- `scheduleAtFixedRate(Runnable,long,long,TimeUnit)`
- `scheduleWithFixedDelay(Runnable,long,long,TimeUnit)`

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

Overview of ScheduledThreadPoolExecutor & Its Policies

- ScheduledExecutorService must be implemented before it is useful (like any interface)
- The Executors utility class defines several factory methods that create implementations

<<Java Class>>	
Executors	
	<code>newFixedThreadPool(int):ExecutorService</code>
	<code>newWorkStealingPool(int):ExecutorService</code>
	<code>newWorkStealingPool():ExecutorService</code>
	<code>newFixedThreadPool(int,ThreadFactory):ExecutorService</code>
	<code>newSingleThreadExecutor():ExecutorService</code>
	<code>newSingleThreadExecutor(ThreadFactory):ExecutorService</code>
	<code>newCachedThreadPool():ExecutorService</code>
	<code>newCachedThreadPool(ThreadFactory):ExecutorService</code>
	<code>newSingleThreadScheduledExecutor():ScheduledExecutorService</code>
	<code>newSingleThreadScheduledExecutor(ThreadFactory):ScheduledExecutorService</code>
	<code>newScheduledThreadPool(int):ScheduledExecutorService</code>
	<code>newScheduledThreadPool(int,ThreadFactory):ScheduledExecutorService</code>
	<code>defaultThreadFactory()</code>
	<code>privilegedThreadFactory()</code>
	<code>callable(Runnable,T):Callable<T></code>
	<code>callable(Runnable):Callable<Object></code>
	<code>callable(PrivilegedAction<?>):Callable<Object></code>
	<code>callable(PrivilegedExceptionAction<?>):Callable<Object></code>
	<code>privilegedCallable(Callable<T>):Callable<T></code>
	<code>privilegedCallableUsingCurrentClassLoader(Callable<T>):Callable<T></code>






















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

Overview of ScheduledThreadPoolExecutor & Its Policies

- ScheduledExecutorService must be implemented before it is useful (like any interface)
 - The Executors utility class defines several factory methods that create implementations
- These implementations are based (directly or indirectly) on the ScheduledThreadPoolExecutor

<<Java Class>>

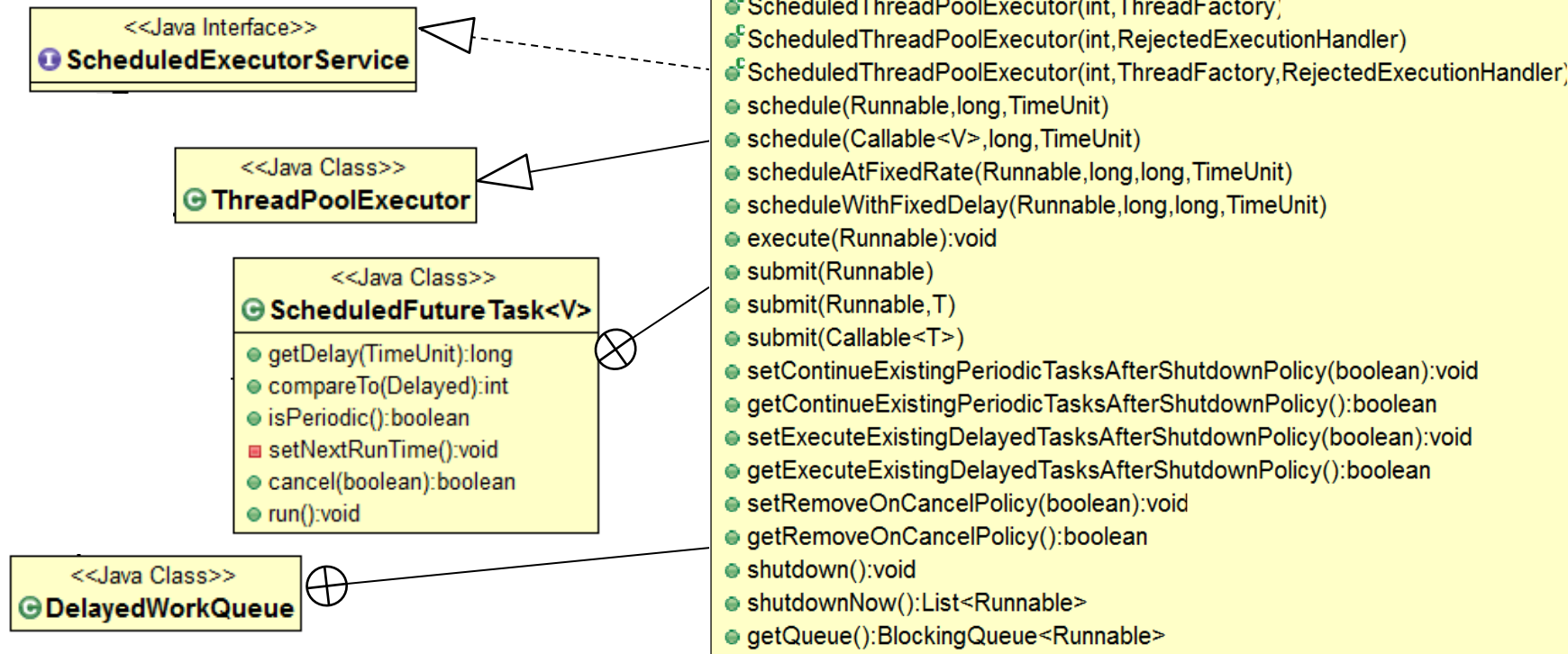
ScheduledThreadPoolExecutor

```
 ScheduledThreadPoolExecutor(int)
 ScheduledThreadPoolExecutor(int, ThreadFactory)
 ScheduledThreadPoolExecutor(int, RejectedExecutionHandler)
 ScheduledThreadPoolExecutor(int, ThreadFactory, RejectedExecutionHandler)
 schedule(Runnable, long, TimeUnit)
 schedule(Callable<V>, long, TimeUnit)
 scheduleAtFixedRate(Runnable, long, long, TimeUnit)
 scheduleWithFixedDelay(Runnable, long, long, TimeUnit)
 execute(Runnable):void
 submit(Runnable)
 submit(Runnable, T)
 submit(Callable<T>)
 setContinueExistingPeriodicTasksAfterShutdownPolicy(boolean):void
 getContinueExistingPeriodicTasksAfterShutdownPolicy():boolean
 setExecuteExistingDelayedTasksAfterShutdownPolicy(boolean):void
 getExecuteExistingDelayedTasksAfterShutdownPolicy():boolean
 setRemoveOnCancelPolicy(boolean):void
 getRemoveOnCancelPolicy():boolean
 shutdown():void
 shutdownNow():List<Runnable>
 getQueue():BlockingQueue<Runnable>
```

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

Overview of ScheduledThreadPoolExecutor & Its Policies

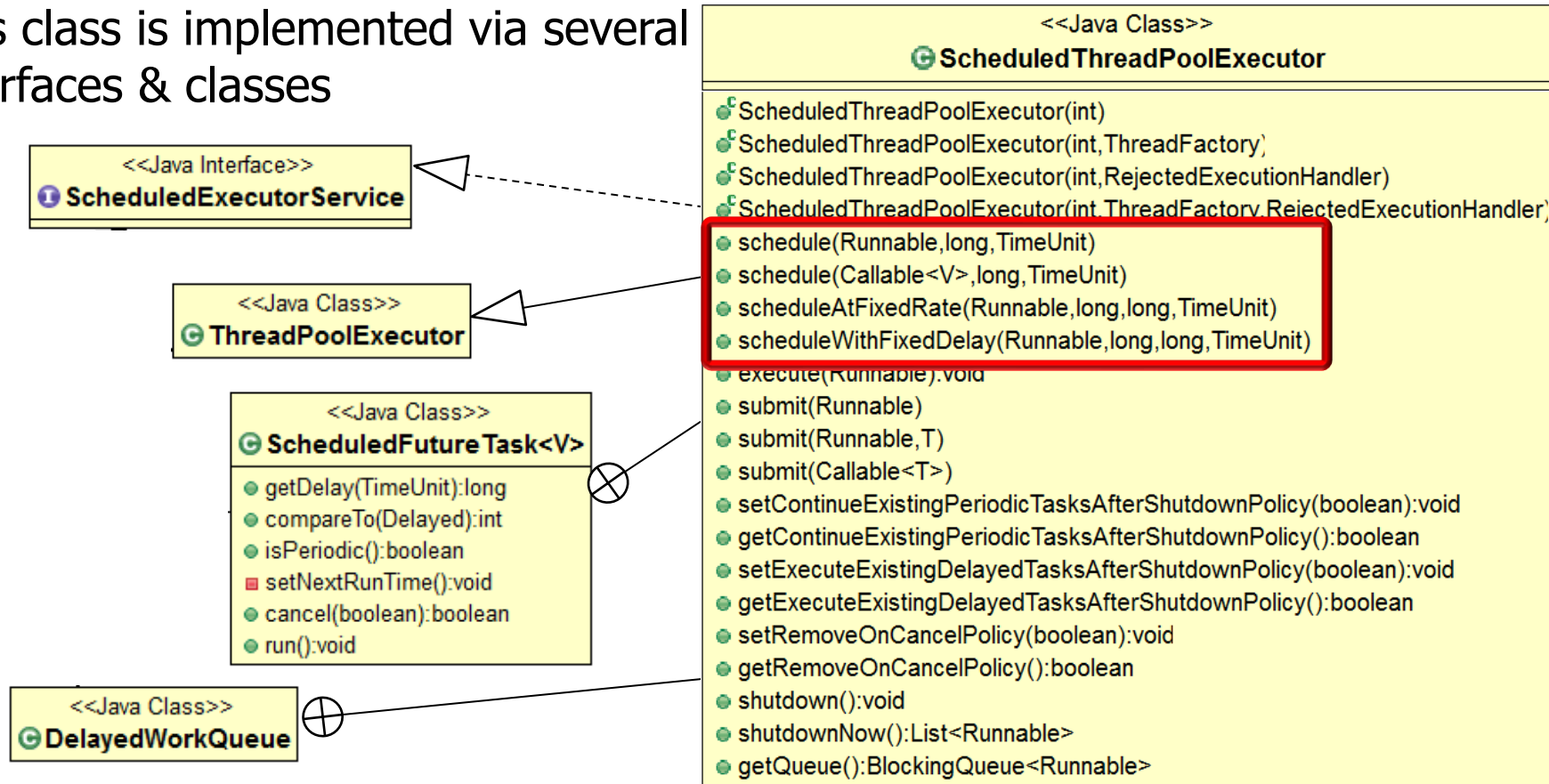
- This class is implemented via several interfaces & classes



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

Overview of ScheduledThreadPoolExecutor & Its Policies

- This class is implemented via several interfaces & classes



Schedules commands that run after a delay and/or executes them periodically

Overview of ScheduledThreadPoolExecutor & Its Policies

- Apps that need to change default policies need to access this class directly

<<Java Class>>	
G ScheduledThreadPoolExecutor	
•	ScheduledThreadPoolExecutor(int)
•	ScheduledThreadPoolExecutor(int, ThreadFactory)
•	ScheduledThreadPoolExecutor(int, RejectedExecutionHandler)
•	ScheduledThreadPoolExecutor(int, ThreadFactory, RejectedExecutionHandler)
•	schedule(Runnable, long, TimeUnit)
•	schedule(Callable<V>, long, TimeUnit)
•	scheduleAtFixedRate(Runnable, long, long, TimeUnit)
•	scheduleWithFixedDelay(Runnable, long, long, TimeUnit)
•	execute(Runnable):void
•	submit(Runnable)
•	submit(Runnable, T)
•	submit(Callable<T>)
•	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 ScheduledThreadPoolExecutor & Its Policies

- Apps that need to change default policies need to access this class directly, e.g.
- Set/get policy to continue running periodic tasks even when executor shuts down (defaults to true)

<<Java Class>>	
G ScheduledThreadPoolExecutor	
•	ScheduledThreadPoolExecutor(int)
•	ScheduledThreadPoolExecutor(int, ThreadFactory)
•	ScheduledThreadPoolExecutor(int, RejectedExecutionHandler)
•	ScheduledThreadPoolExecutor(int, ThreadFactory, RejectedExecutionHandler)
•	schedule(Runnable, long, TimeUnit)
•	schedule(Callable<V>, long, TimeUnit)
•	scheduleAtFixedRate(Runnable, long, long, TimeUnit)
•	scheduleWithFixedDelay(Runnable, long, long, TimeUnit)
•	execute(Runnable):void
•	submit(Runnable)
•	submit(Runnable, T)
•	submit(Callable<T>)
•	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 ScheduledThreadPoolExecutor & Its Policies

- Apps that need to change default policies need to access this class directly, e.g.
 - Set/get policy to continue running periodic tasks even when executor shuts down (defaults to true)
 - Set/get policy to run delayed tasks even when the executor has been shutdown (defaults to true)

<<Java Class>>	
G ScheduledThreadPoolExecutor	
•	ScheduledThreadPoolExecutor(int)
•	ScheduledThreadPoolExecutor(int, ThreadFactory)
•	ScheduledThreadPoolExecutor(int, RejectedExecutionHandler)
•	ScheduledThreadPoolExecutor(int, ThreadFactory, RejectedExecutionHandler)
•	schedule(Runnable, long, TimeUnit)
•	schedule(Callable<V>, long, TimeUnit)
•	scheduleAtFixedRate(Runnable, long, long, TimeUnit)
•	scheduleWithFixedDelay(Runnable, long, long, TimeUnit)
•	execute(Runnable):void
•	submit(Runnable)
•	submit(Runnable, T)
•	submit(Callable<T>)
•	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 ScheduledThreadPoolExecutor & Its Policies

- Apps that need to change default policies need to access this class directly, e.g.
 - Set/get policy to continue running periodic tasks even when executor shuts down (defaults to true)
 - Set/get policy to run delayed tasks even when the executor has been shutdown (defaults to true)
 - Set/gets the policy to immediately remove tasks from work queue when they are cancelled (defaults to false)

<<Java Class>>	
G ScheduledThreadPoolExecutor	
G	ScheduledThreadPoolExecutor(int)
G	ScheduledThreadPoolExecutor(int, ThreadFactory)
G	ScheduledThreadPoolExecutor(int, RejectedExecutionHandler)
G	ScheduledThreadPoolExecutor(int, ThreadFactory, RejectedExecutionHandler)
G	schedule(Runnable, long, TimeUnit)
G	schedule(Callable<V>, long, TimeUnit)
G	scheduleAtFixedRate(Runnable, long, long, TimeUnit)
G	scheduleWithFixedDelay(Runnable, long, long, TimeUnit)
G	execute(Runnable):void
G	submit(Runnable)
G	submit(Runnable, T)
G	submit(Callable<T>)
G	setContinueExistingPeriodicTasksAfterShutdownPolicy(boolean):void
G	getContinueExistingPeriodicTasksAfterShutdownPolicy():boolean
G	setExecuteExistingDelayedTasksAfterShutdownPolicy(boolean):void
G	getExecuteExistingDelayedTasksAfterShutdownPolicy():boolean
G	setRemoveOnCancelPolicy(boolean):void
G	getRemoveOnCancelPolicy():boolean
G	shutdown():void
G	shutdownNow():List<Runnable>
G	getQueue():BlockingQueue<Runnable>

Overview of ScheduledThreadPoolExecutor & Its Policies

- Apps that need to change default policies need to access this class directly



<<Java Class>>	
G ScheduledThreadPoolExecutor	
G	ScheduledThreadPoolExecutor(int)
G	ScheduledThreadPoolExecutor(int, ThreadFactory)
G	ScheduledThreadPoolExecutor(int, RejectedExecutionHandler)
G	ScheduledThreadPoolExecutor(int, ThreadFactory, RejectedExecutionHandler)
G	schedule(Runnable, long, TimeUnit)
G	schedule(Callable<V>, long, TimeUnit)
G	scheduleAtFixedRate(Runnable, long, long, TimeUnit)
G	scheduleWithFixedDelay(Runnable, long, long, TimeUnit)
G	execute(Runnable):void
G	submit(Runnable)
G	submit(Runnable, T)
G	submit(Callable<T>)
G	setContinueExistingPeriodicTasksAfterShutdownPolicy(boolean):void
G	getContinueExistingPeriodicTasksAfterShutdownPolicy():boolean
G	setExecuteExistingDelayedTasksAfterShutdownPolicy(boolean):void
G	getExecuteExistingDelayedTasksAfterShutdownPolicy():boolean
G	setRemoveOnCancelPolicy(boolean):void
G	getRemoveOnCancelPolicy():boolean
G	shutdown():void
G	shutdownNow():List<Runnable>
G	getQueue():BlockingQueue<Runnable>

Oddly, all three of these policies are set "backwards" by default..

Overview of ScheduledThreadPoolExecutor & Its Policies

- Don't try to access a ScheduledThreadPoolExecutor by casting the result of Executors.newSingleThreadScheduledExecutor()

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



Overview of ScheduledThreadPoolExecutor & Its Policies

- Don't try to access a ScheduledThreadPoolExecutor by casting the result of Executors.newSingleThreadScheduledExecutor()

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

newSingleThreadScheduledExecutor() doesn't return ScheduledThreadPoolExecutor!

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

See [java/util/concurrent/Executors.java#Executors.newSingleThreadScheduledExecutor](http://java.util.concurrent.Executors.java#Executors.newSingleThreadScheduledExecutor)

Overview of ScheduledThreadPoolExecutor & Its Policies

- Therefore, if you need a single-threaded ScheduledThreadPoolExecutor with non-default policies you'll need to make one yourself

```
ScheduledThreadPoolExecutor exec =  
    new ScheduledThreadPoolExecutor(1);
```

or

```
ScheduledThreadPoolExecutor exec = (ScheduledThreadPoolExecutor)  
    Executors.newScheduledThreadPool(1);
```

```
exec.setRemoveOnCancelPolicy(true);  
exec.setContinueExistingPeriodicTasksAfterShutdownPolicy(false);  
exec.setExecuteExistingDelayedTasksAfterShutdownPolicy(false);
```

See lesson on "*The Java ScheduledExecutorService: Implementing TimedMemoizerEx*"

Overview of ScheduledThreadPoolExecutor & Its Policies

- Therefore, if you need a single-threaded ScheduledThreadPoolExecutor with non-default policies you'll need to make one yourself

```
ScheduledThreadPoolExecutor exec =  
    new ScheduledThreadPoolExecutor(1);
```

or

Create a single-threaded ScheduledThreadPoolExecutor

```
ScheduledThreadPoolExecutor exec = (ScheduledThreadPoolExecutor)  
    Executors.newScheduledThreadPool(1);
```

```
exec.setRemoveOnCancelPolicy(true);  
exec.setContinueExistingPeriodicTasksAfterShutdownPolicy(false);  
exec.setExecuteExistingDelayedTasksAfterShutdownPolicy(false);
```

See knpcode.com/java/concurrency/java-scheduledthreadpoolexecutor-scheduling-with-executorservice

Overview of ScheduledThreadPoolExecutor & Its Policies

- Therefore, if you need a single-threaded ScheduledThreadPoolExecutor with non-default policies you'll need to make one yourself

```
ScheduledThreadPoolExecutor exec =  
    new ScheduledThreadPoolExecutor(1);
```

or

```
ScheduledThreadPoolExecutor exec = (ScheduledThreadPoolExecutor)  
    Executors.newScheduledThreadPool(1);
```

Set default policies to something more sensible

```
exec.setRemoveOnCancelPolicy(true);  
exec.setContinueExistingPeriodicTasksAfterShutdownPolicy(false);  
exec.setExecuteExistingDelayedTasksAfterShutdownPolicy(false);
```

End of Overview of Java ScheduledExecutor Service Policies