

The Java ExecutorService Interface

(Part 1)

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 powerful features defined in the Java `ExecutorService` interface & related interfaces/classes

Interface `ExecutorService`

All Superinterfaces:

`Executor`

All Known Subinterfaces:

`ScheduledExecutorService`

All Known Implementing Classes:

`AbstractExecutorService`, `ForkJoinPool`,
`ScheduledThreadPoolExecutor`, `ThreadPoolExecutor`

```
public interface ExecutorService  
    extends Executor
```

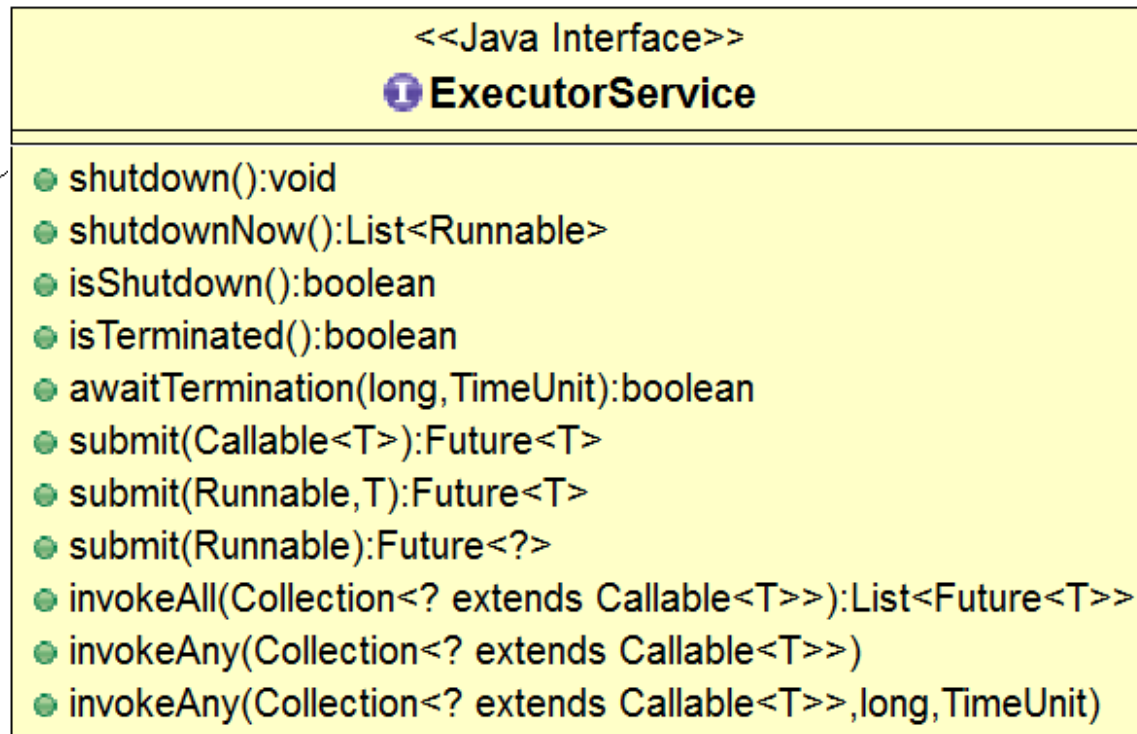
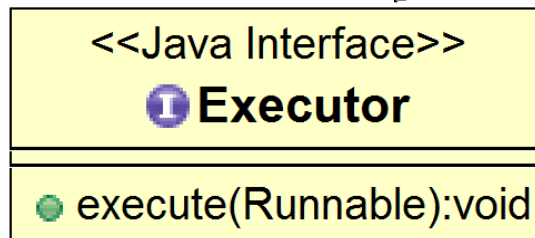
An `Executor` that provides methods to manage termination and methods that can produce a `Future` for tracking progress of one or more asynchronous tasks.

An `ExecutorService` can be shut down, which will cause it to reject new tasks. Two different methods are provided for shutting down an `ExecutorService`. The `shutdown()` method will allow previously submitted tasks to execute before terminating, while the `shutdownNow()` method prevents waiting tasks from starting and attempts to stop currently executing tasks. Upon termination, an executor has no tasks actively executing, no tasks awaiting execution, and no new tasks can be submitted. An unused `ExecutorService` should be shut down to allow reclamation of its resources.

Overview of the ExecutorService Interface

Overview of the ExecutorService Interface

- Extends Executor



Overview of the ExecutorService Interface

- Extends Executor
 - Submit tasks & return futures for these tasks

<<Java Interface>>

ExecutorService

- shutdown():void
- shutdownNow():List<Runnable>
- isShutdown():boolean
- isTerminated():boolean
- awaitTermination(long, TimeUnit):boolean
- submit(Callable<T>):Future<T>
- submit(Runnable, T):Future<T>
- submit(Runnable):Future<?>
- invokeAll(Collection<? extends Callable<T>>):List<Future<T>>
- invokeAny(Collection<? extends Callable<T>>)
- invokeAny(Collection<? extends Callable<T>>, long, TimeUnit)

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

Overview of the ExecutorService Interface

- Extends Executor
 - Submit tasks & return futures for these tasks
- Manage lifecycle of tasks & worker threads

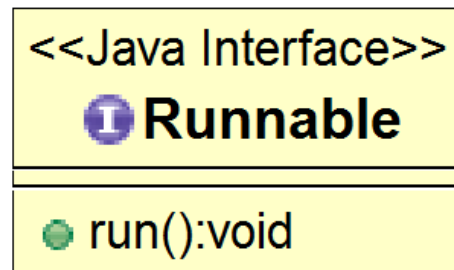
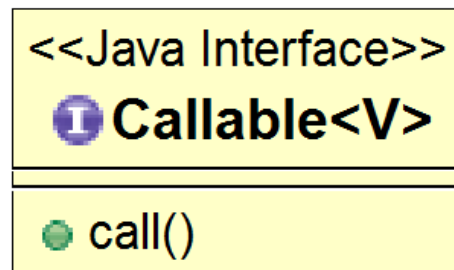
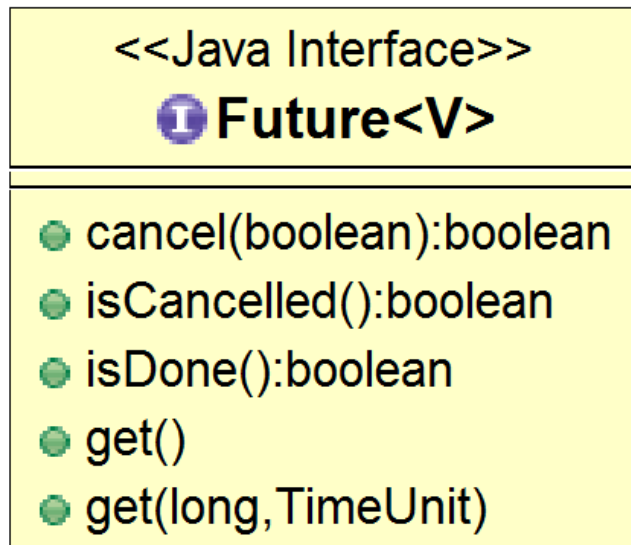
<<Java Interface>>

ExecutorService

- shutdown():void
- shutdownNow():List<Runnable>
- isShutdown():boolean
- isTerminated():boolean
- awaitTermination(long, TimeUnit):boolean
- submit(Callable<T>):Future<T>
- submit(Runnable, T):Future<T>
- submit(Runnable):Future<?>
- invokeAll(Collection<? extends Callable<T>>):List<Future<T>>
- invokeAny(Collection<? extends Callable<T>>)
- invokeAny(Collection<? extends Callable<T>>, long, TimeUnit)

Overview of the ExecutorService Interface

- The ExecutorService is used with other interfaces

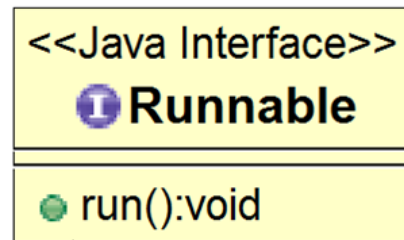


Overview of the ExecutorService Interface

- The ExecutorService is used with other interfaces, e.g.

- **Runnable**

- A “one-way” task that does not return a result



Runnable is a functional interface

Overview of the ExecutorService Interface

- The ExecutorService is used with other interfaces, e.g.

- Runnable**

- Callable**

- A “two-way” task that returns a result



<<Java Interface>>

Callable<V>

call()

Callable is also a functional interface

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

Overview of the ExecutorService Interface

- The ExecutorService is used with other interfaces, e.g.

- Runnable**

- Callable**

- A “two-way” task that returns a result
- Typically used to run two-way async tasks

<<Java Interface>>

Callable<V>

call()



Overview of the ExecutorService Interface

- The ExecutorService is used with other interfaces, e.g.

- Runnable**

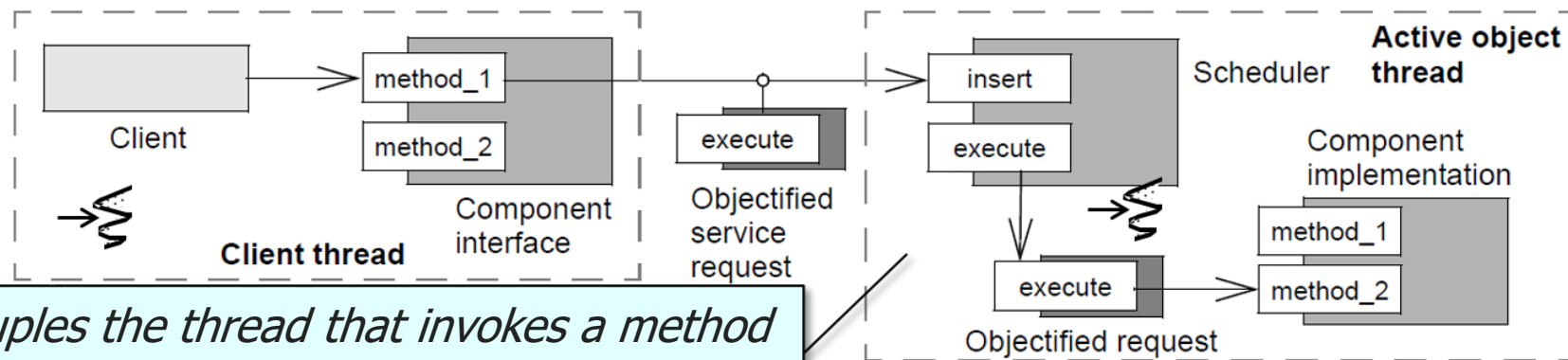
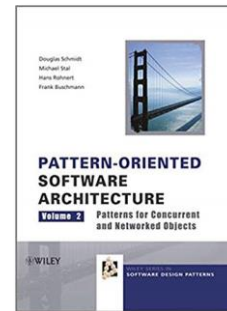
- Callable**

- A “two-way” task that returns a result
- Typically used to run two-way async tasks
- Implements the *Active Object* pattern

<<Java Interface>>

Callable<V>

call()



Decouples the thread that invokes a method from the thread that executes the method

See en.wikipedia.org/wiki/Active_object

Overview of the ExecutorService Interface

- The ExecutorService is used with other interfaces, e.g.
 - **Runnable**
 - **Callable**
 - **Future**
 - Represents async two-way task result

<<Java Interface>>

Future<V>

- cancel(boolean):boolean
- isCancelled():boolean
- isDone():boolean
- get()
- get(long,TimeUnit)



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

Overview of the ExecutorService Interface

- The ExecutorService is used with other interfaces, e.g.
 - **Runnable**
 - **Callable**
 - **Future**
 - Represents async two-way task result
 - Can be canceled & tested for completion

<<Java Interface>>	
❏ Future<V>	
●	cancel(boolean):boolean
●	isCancelled():boolean
●	isDone():boolean
●	get()
●	get(long,TimeUnit)



Overview of the ExecutorService Interface

- The ExecutorService is used with other interfaces, e.g.

- **Runnable**

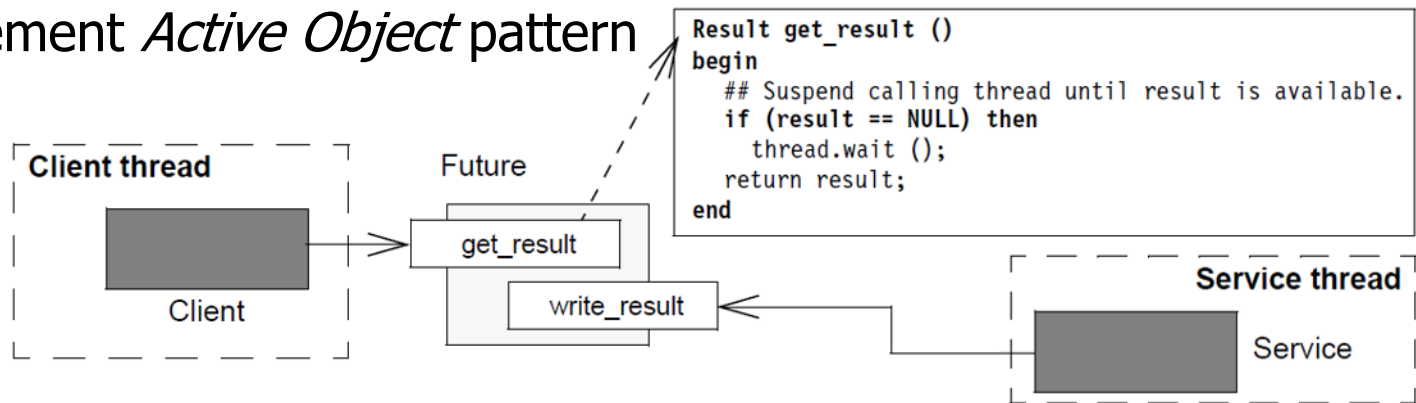
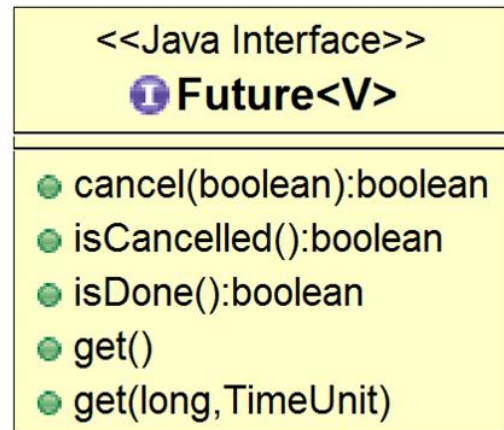
- **Callable**

- **Future**

- Represents async two-way task result

- Can be canceled & tested for completion

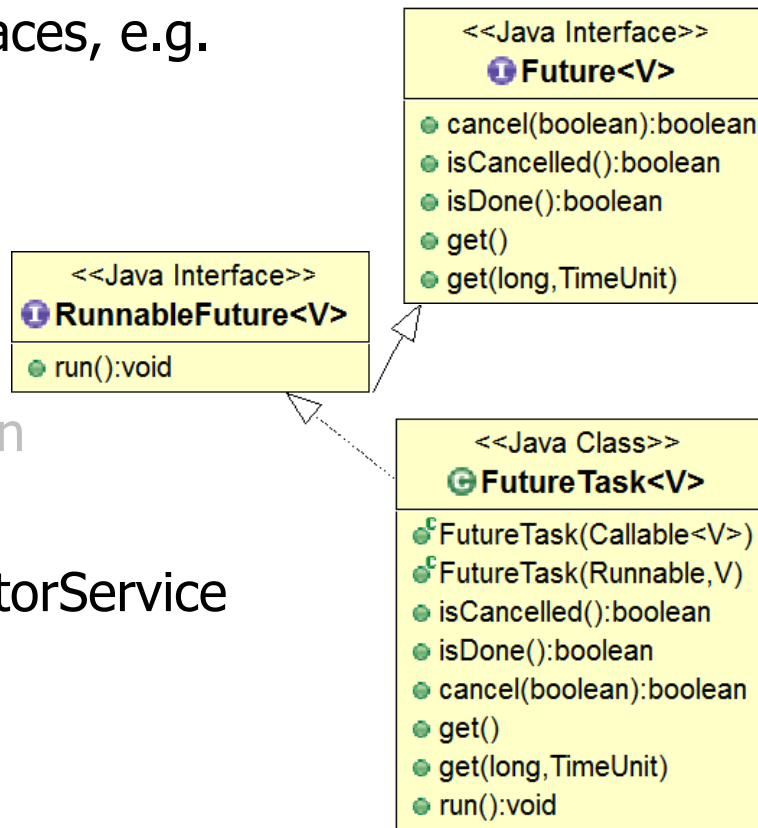
- Can implement *Active Object* pattern



See en.wikipedia.org/wiki/Active_object

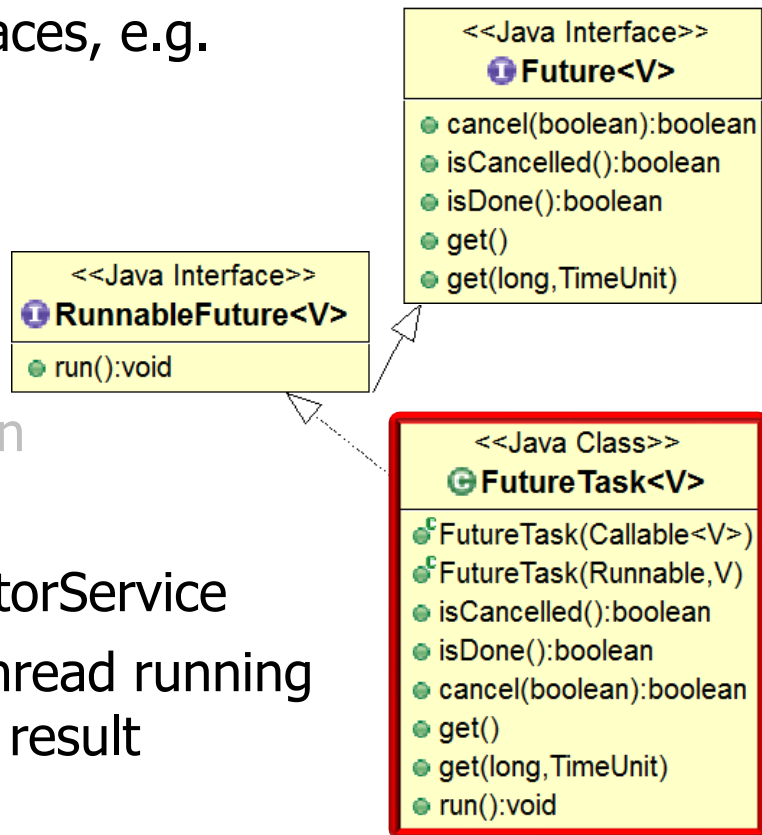
Overview of the ExecutorService Interface

- The ExecutorService is used with other interfaces, e.g.
 - **Runnable**
 - **Callable**
 - **Future**
 - Represents async two-way task result
 - Can be canceled & tested for completion
 - Can implement *Active Object* pattern
 - Other Future variants implement ExecutorService



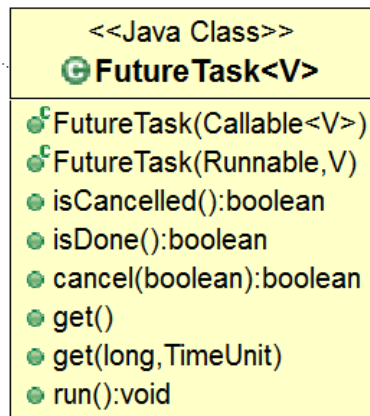
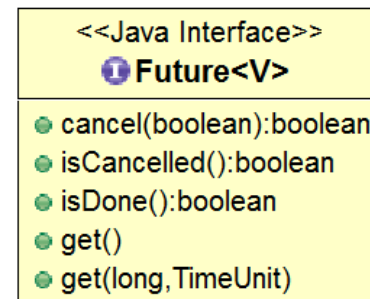
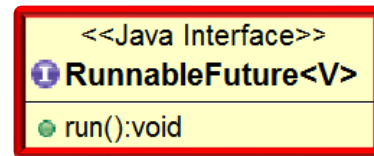
Overview of the ExecutorService Interface

- The ExecutorService is used with other interfaces, e.g.
 - **Runnable**
 - **Callable**
 - **Future**
 - Represents async two-way task result
 - Can be canceled & tested for completion
 - Can implement *Active Object* pattern
 - Other Future variants implement ExecutorService
 - **FutureTask** – Conveys result from thread running a computation to thread(s) retrieving result



Overview of the ExecutorService Interface

- The ExecutorService is used with other interfaces, e.g.
 - **Runnable**
 - **Callable**
 - **Future**
 - Represents async two-way task result
 - Can be canceled & tested for completion
 - Can implement *Active Object* pattern
 - Other Future variants implement ExecutorService
 - **FutureTask**
 - **RunnableFuture** – Execution of run() method will complete the future & allow access to its results



Overview of the ExecutorService Interface

- The ExecutorService is used with other interfaces, e.g.

- Runnable
- Callable
- Future
- CompletableFuture**
 - This Future variant supports dependent actions that are trigger upon its completion

CompletableFuture isn't part of the Java Executor framework

```
<<Java Class>>
G CompletableFuture<T>

c CompletableFuture()
s supplyAsync(Supplier<U>):CompletableFuture<U>
s supplyAsync(Supplier<U>,Executor):CompletableFuture<U>
s runAsync(Runnable):CompletableFuture<Void>
s runAsync(Runnable,Executor):CompletableFuture<Void>
s completedFuture(U):CompletableFuture<U>
  isDone():boolean
  get()
  get(long,TimeUnit)
  join()
  complete(T):boolean
  thenApply(Function<?>):CompletableFuture<U>
  thenAccept(Consumer<? super T>):CompletableFuture<Void>
  thenCombine(CompletionStage<? extends U>,BiFunction<?>):CompletableFuture<V>
  thenCompose(Function<?>):CompletableFuture<U>
  whenComplete(BiConsumer<?>):CompletableFuture<T>
s allOf(CompletableFuture[]<?>):CompletableFuture<Void>
s anyOf(CompletableFuture[]<?>):CompletableFuture<Object>
  isCancelled():boolean
```

Overview of the ExecutorService Interface

- The ExecutorService is used with other interfaces, e.g.

- Runnable
- Callable
- Future
- CompletableFuture**

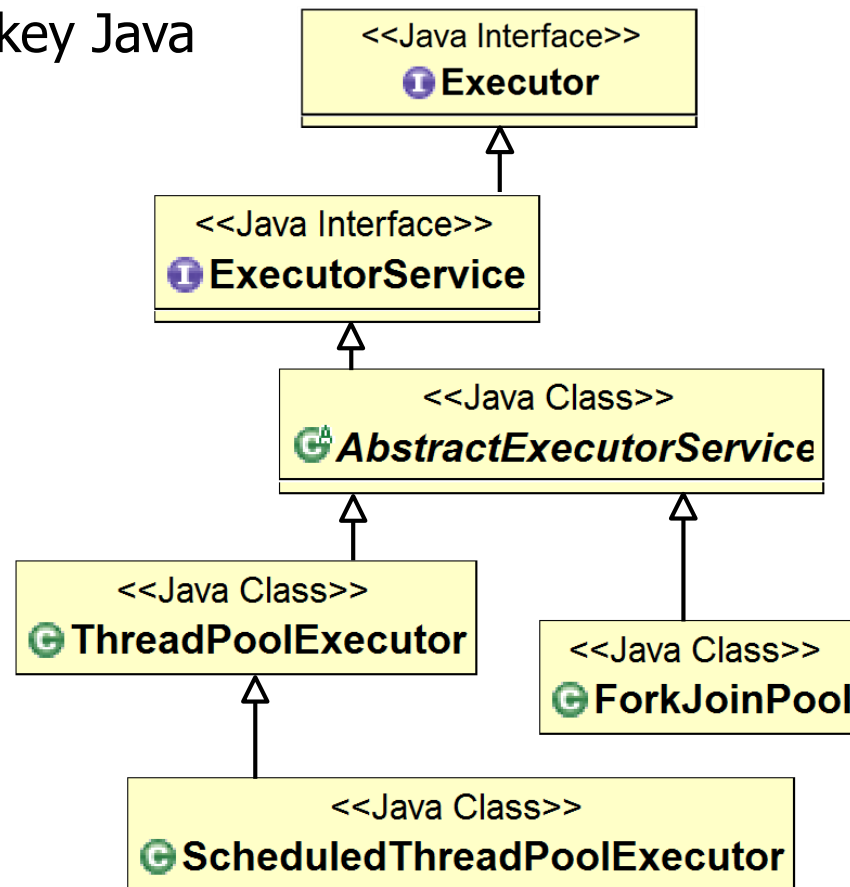
- This Future variant supports dependent actions that are trigger upon its completion
- This topic is covered in other courses

```
<<Java Class>>
G CompletableFuture<T>

c CompletableFuture()
s supplyAsync(Supplier<U>):CompletableFuture<U>
s supplyAsync(Supplier<U>,Executor):CompletableFuture<U>
s runAsync(Runnable):CompletableFuture<Void>
s runAsync(Runnable,Executor):CompletableFuture<Void>
s completedFuture(U):CompletableFuture<U>
  isDone():boolean
  get()
  get(long,TimeUnit)
  join()
  complete(T):boolean
  thenApply(Function<?>):CompletableFuture<U>
  thenAccept(Consumer<? super T>):CompletableFuture<Void>
  thenCombine(CompletionStage<? extends U>,BiFunction<?>):CompletableFuture<V>
  thenCompose(Function<?>):CompletableFuture<U>
  whenComplete(BiConsumer<?>):CompletableFuture<T>
s allOf(CompletableFuture[]<?>):CompletableFuture<Void>
s anyOf(CompletableFuture[]<?>):CompletableFuture<Object>
  isCancelled():boolean
```

Overview of the ExecutorService Interface

- ExecutorService also forms the basis for key Java Executor framework subclasses



See <src/share/classes/java/util/concurrent>

End of Overview of Java ExecutorService (Part 1)