Java ExecutorService: Related Interfaces

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

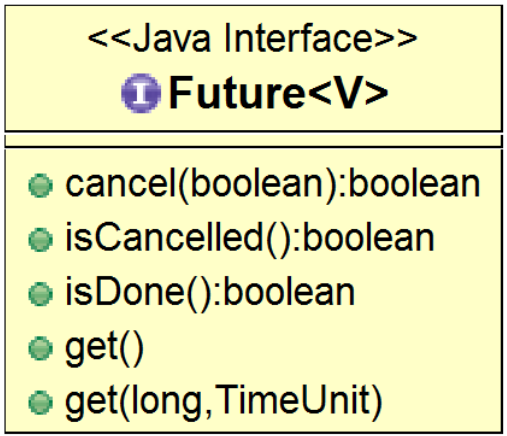
Professor of Computer Science

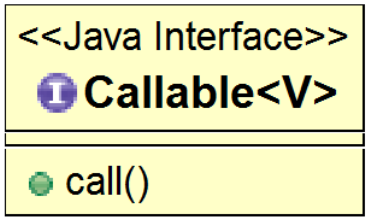
Institute for Software
Integrated Systems

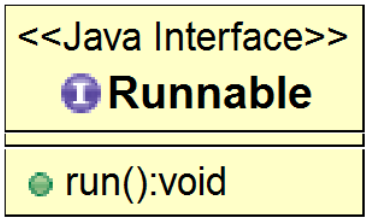
Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Recognize the powerful features defined in the Java ExecutorService interface
- Understand other interfaces related to ExecutorService







Interface ExecutorService

All Superinterfaces:

`Executor`

All Known Subinterfaces:

`ScheduledExecutorService`

All Known Implementing Classes:

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

```
public interface ExecutorService
extends Executor
```

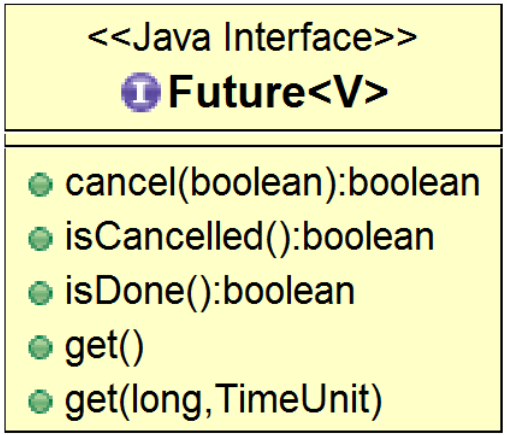
An `Executor` that provides methods to manage termination and methods that can produce a `Future` for tracking progress of one or more asynchronous tasks.

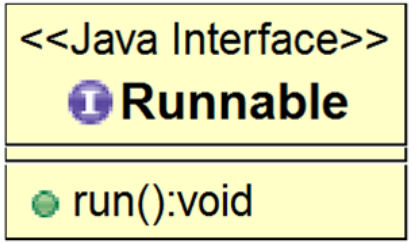
An `ExecutorService` can be shut down, which will cause it to reject new tasks. Two different methods are provided for shutting down an `ExecutorService`. The `shutdown()` method will allow previously submitted tasks to execute before terminating, while the `shutdownNow()` method prevents waiting tasks 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.

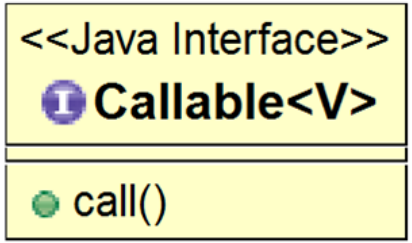
Overview of Interfaces Related To ExecutorService

Overview of Interfaces Related to ExecutorService

- ExecutorService uses several other interfaces to manage task lifecycles

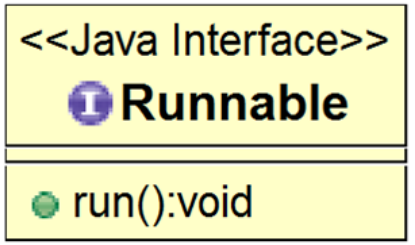


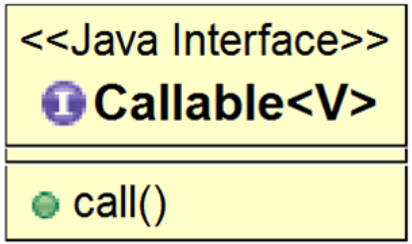




Overview of Interfaces Related to ExecutorService

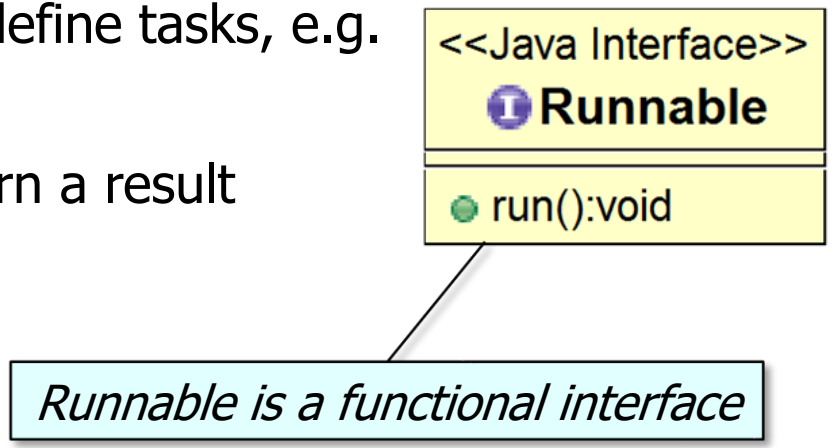
- ExecutorService uses two interfaces to define tasks





Overview of Interfaces Related to ExecutorService

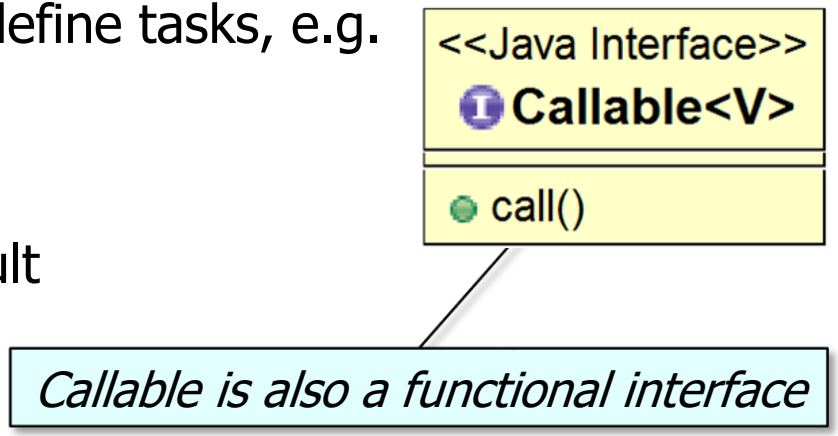
- ExecutorService uses two interfaces to define tasks, e.g.
  - **`Runnable`**
    - A "one-way" task that does not return a result



See docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html

Overview of Interfaces Related to ExecutorService

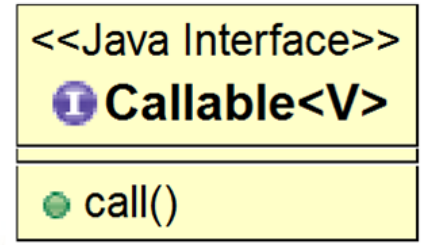
- ExecutorService uses two interfaces to define tasks, e.g.
  - `Runnable`
  - **`Callable`**
    - A "two-way" task that returns a result



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

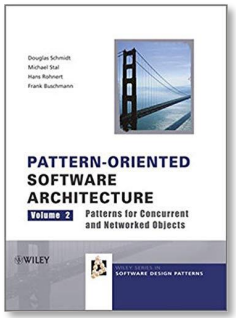
Overview of Interfaces Related to ExecutorService

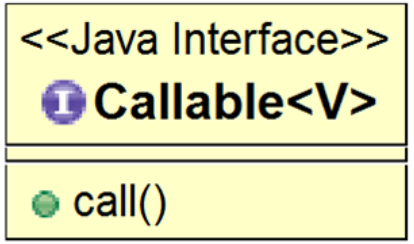
- ExecutorService uses two interfaces to define tasks, e.g.
  - `Runnable`
  - **`Callable`**
    - A "two-way" task that returns a result
    - Typically used to run two-way async tasks

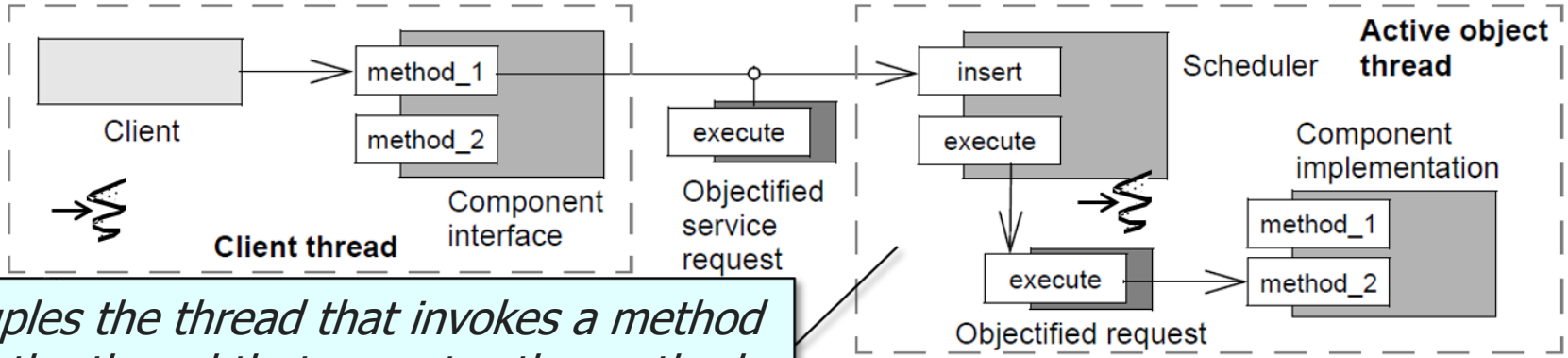


Overview of Interfaces Related to ExecutorService

- ExecutorService uses two interfaces to define tasks, 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







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

See en.wikipedia.org/wiki/Active_object

Overview of Interfaces Related to ExecutorService

- ExecutorService uses the **`Future`** interface to represent & control a task's lifecycle

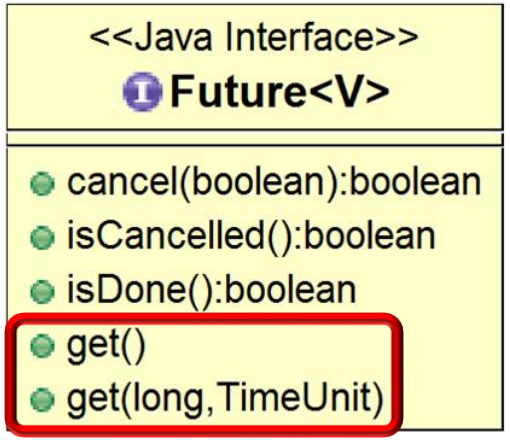
<<Java Interface>>
Future<V>

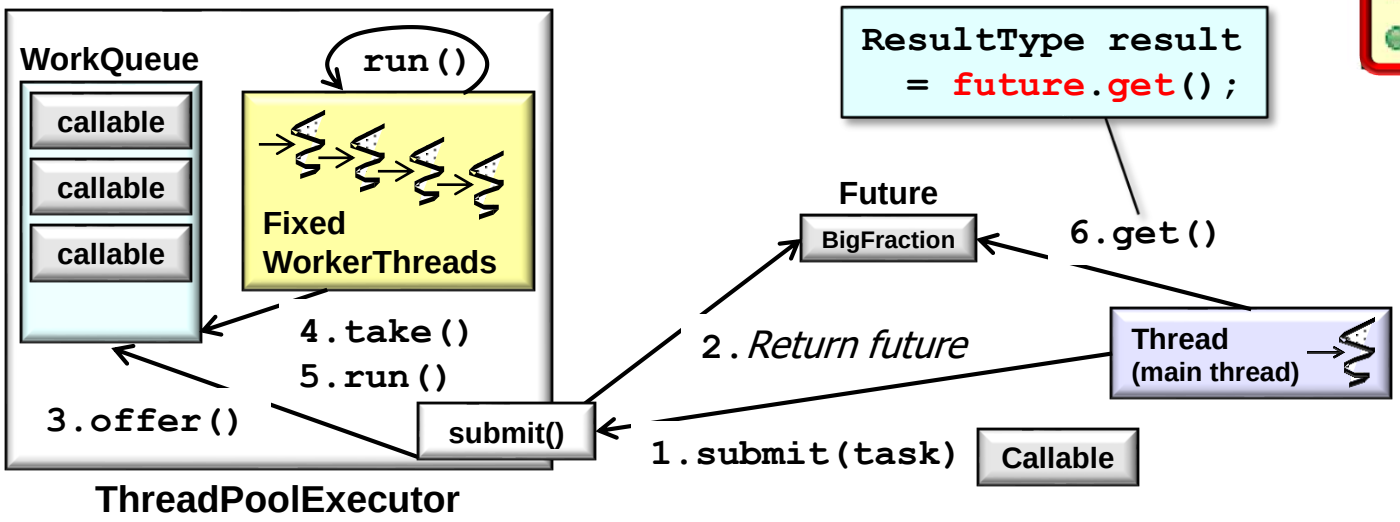
- cancel(boolean):boolean
- isCancelled():boolean
- isDone():boolean
- get()
- get(long,TimeUnit)

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

Overview of Interfaces Related to ExecutorService

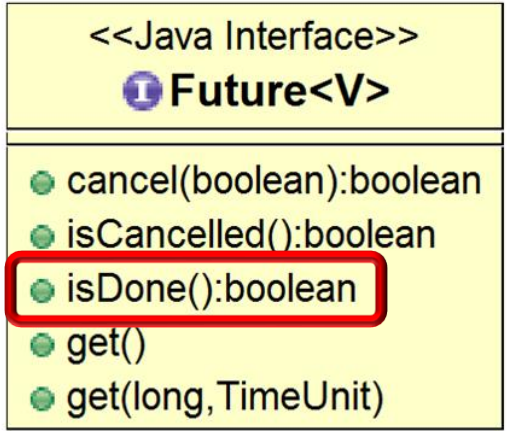
- ExecutorService uses the **`Future`** interface to represent & control a task's lifecycle, e.g.,
  - Can be used to retrieve a two-way task's result

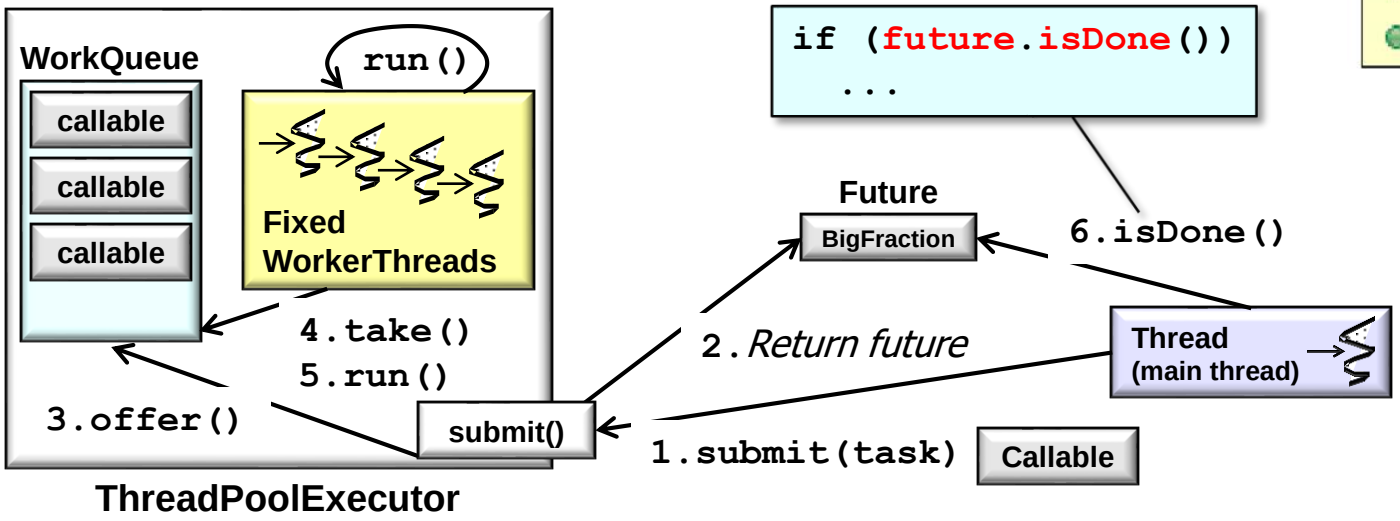




Overview of Interfaces Related to ExecutorService

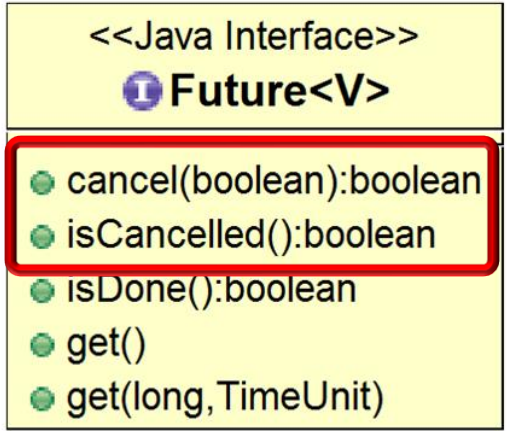
- ExecutorService uses the **`Future`** interface to represent & control a task's lifecycle, e.g.,
  - Can be used to retrieve a two-way task's result
  - Can be tested for completion

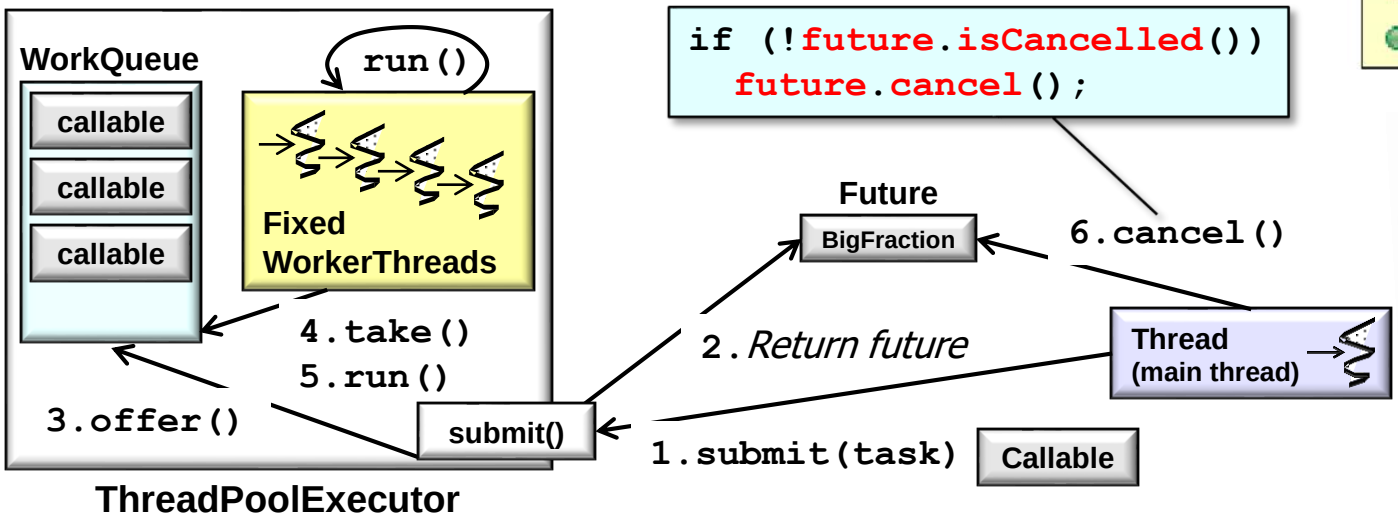




Overview of Interfaces Related to ExecutorService

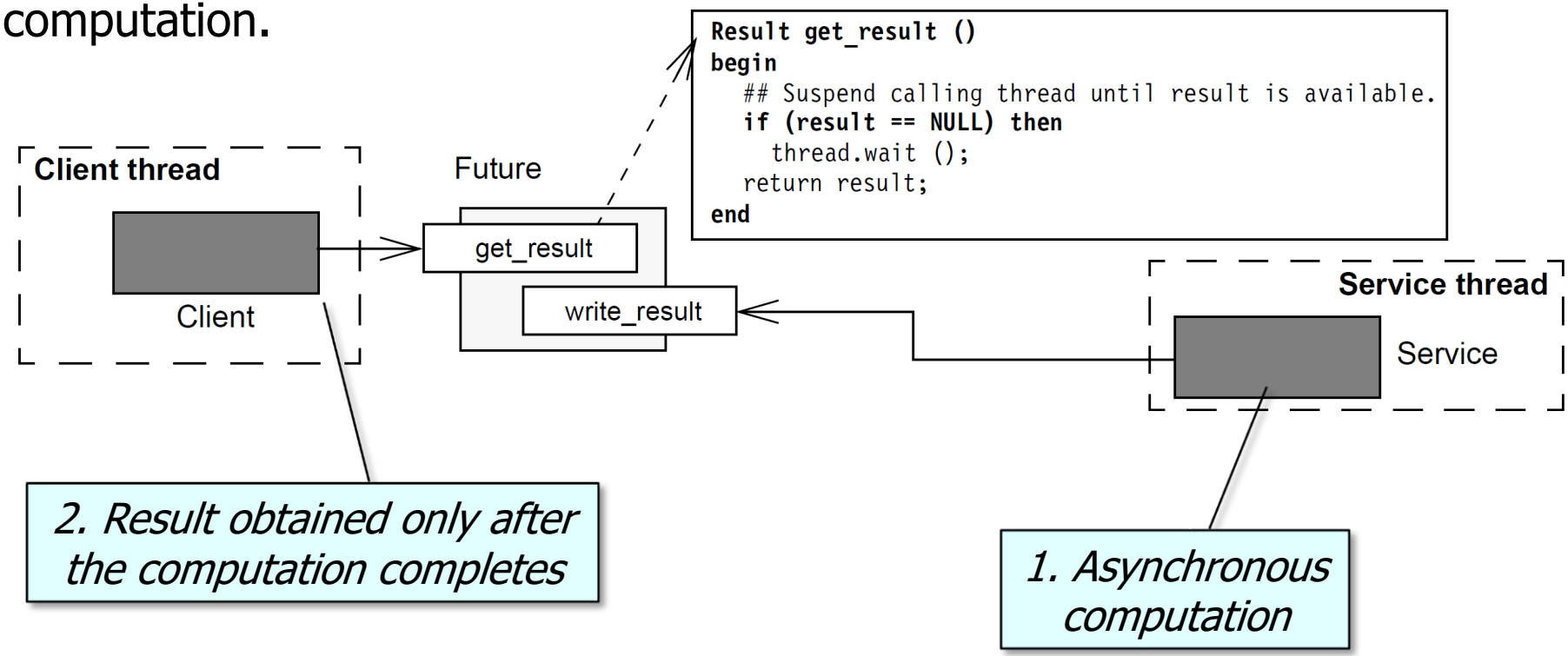
- ExecutorService uses the **`Future`** interface to represent & control a task's lifecycle, e.g.,
  - Can be used to retrieve a two-way task's result
  - Can be tested for completion
  - Can be tested for cancellation & cancelled





Overview of Interfaces Related to ExecutorService

- A Java future defines a proxy to represent & control the result of an async computation.



See en.wikipedia.org/wiki/Futures_and_promises

Overview of Interfaces Related to ExecutorService

- A Java future defines a proxy to represent & control the result of an async computation.

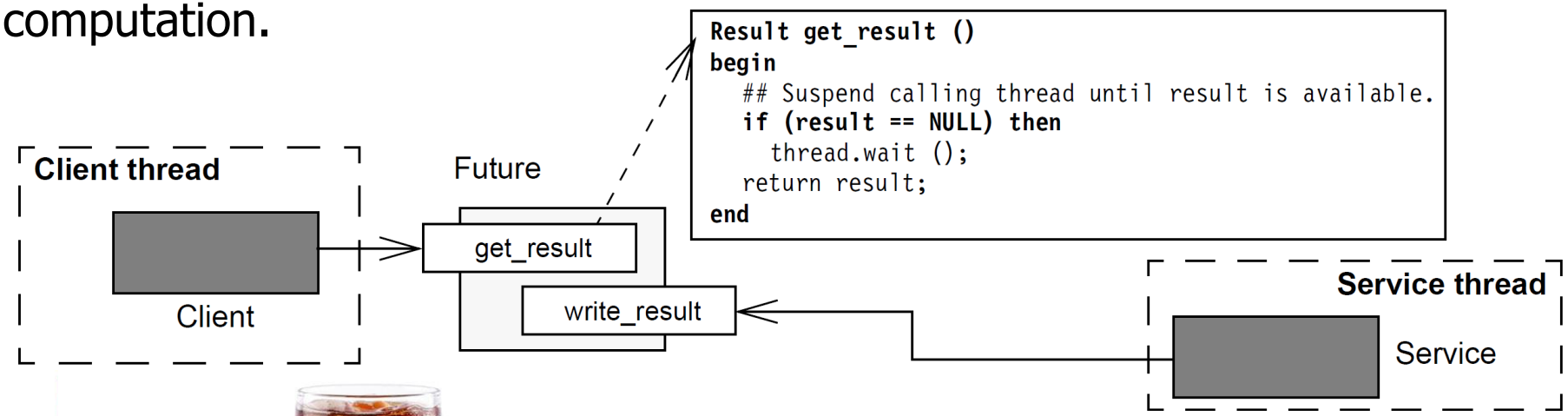


Table tent #'s are a human-known-use of futures!

See www.citygrafx.com/table-numbers-table-markers

Overview of Interfaces Related to ExecutorService

- A Java future defines a proxy to represent & control the result of an async computation.

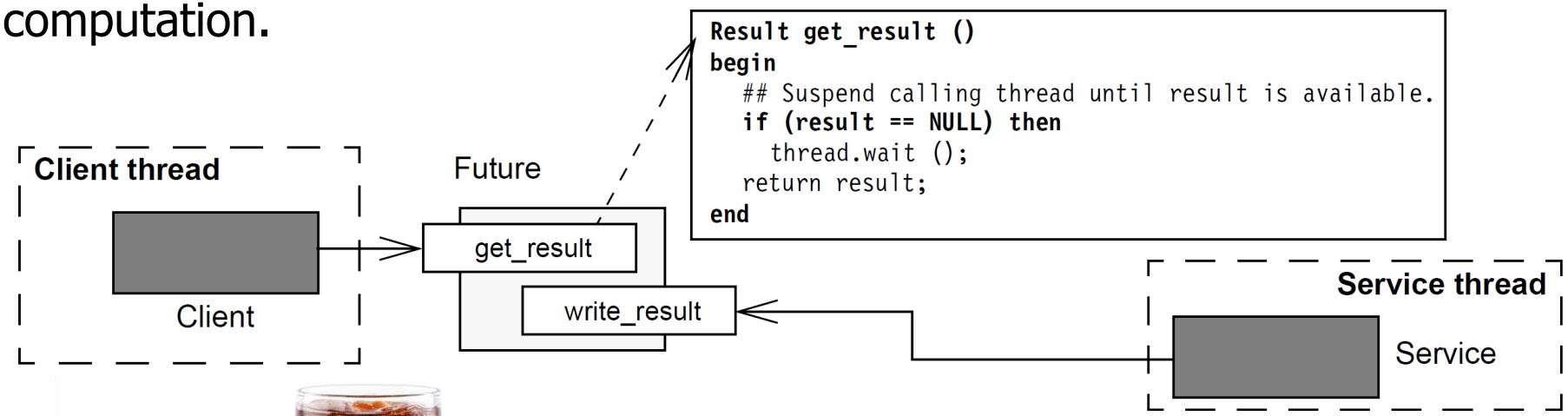
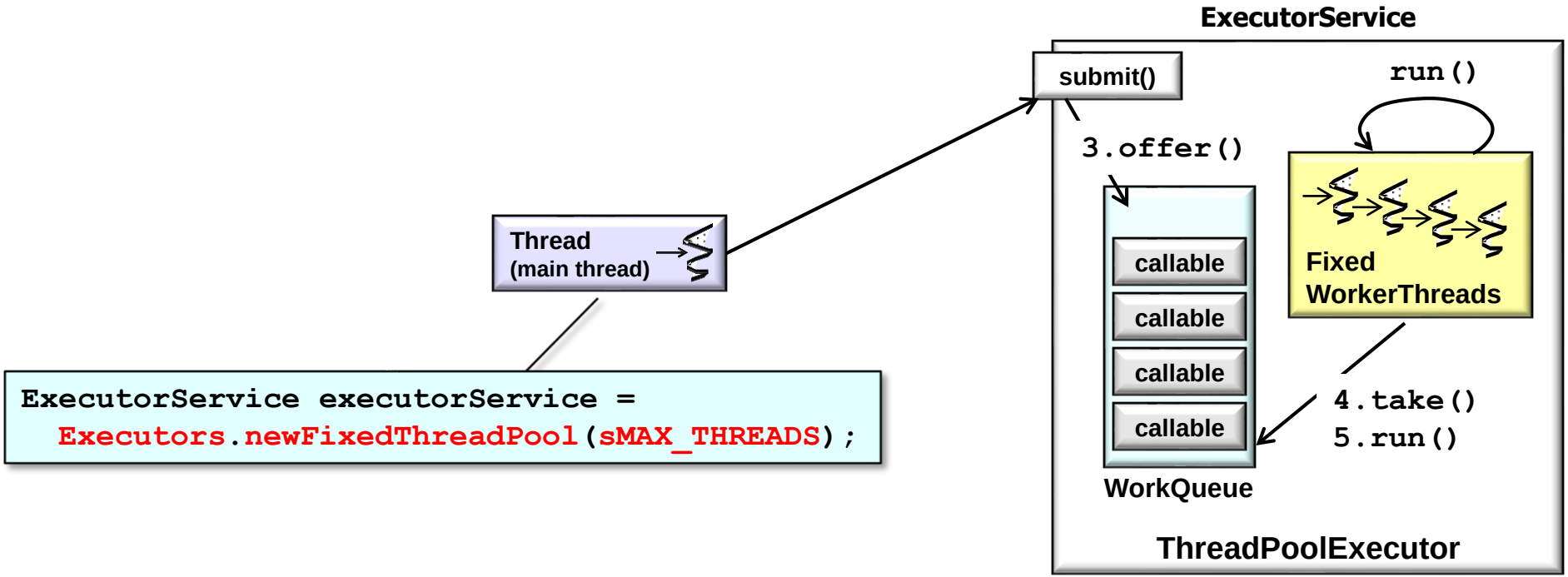


Table tent #'s are a human-known-use of futures!

e.g., McDonald's vs Wendy's model of preparing fast food

Overview of Interfaces Related to ExecutorService

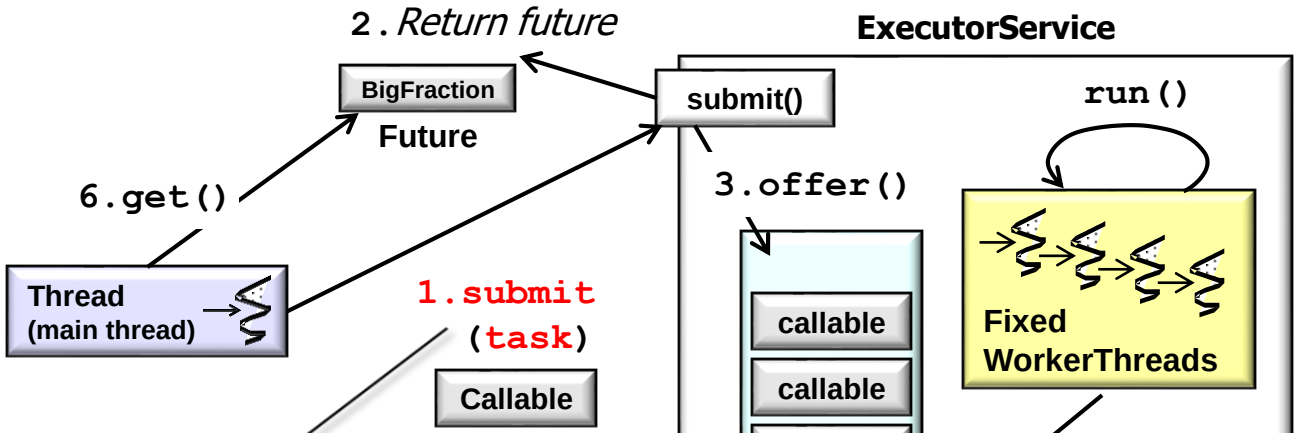
- ExecutorService.submit() can initiate an async computation in Java.



See next part of this lesson on "*Java ExecutorService: Key Methods*"

Overview of Interfaces Related to ExecutorService

- ExecutorService.submit() can initiate an async computation in Java.

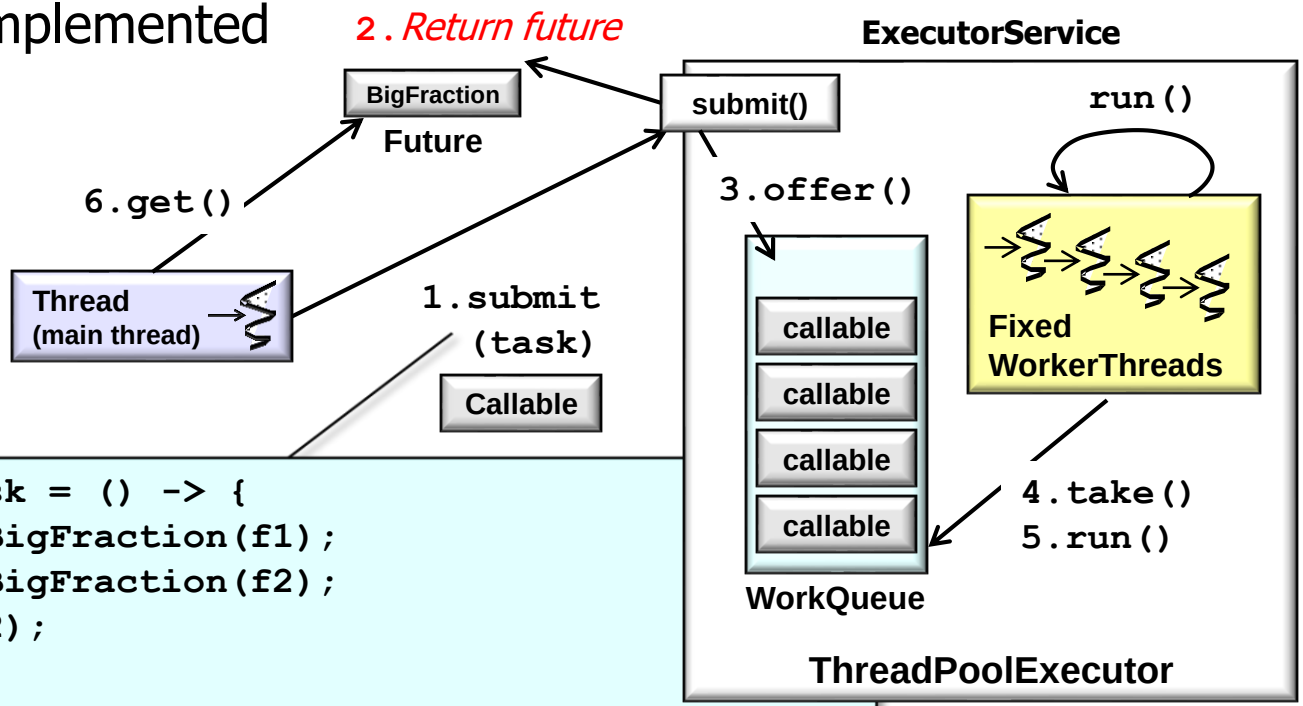



```
Callable<BigFraction> task = () -> {
  BigFraction bf1 = new BigFraction(f1);
  BigFraction bf2 = new BigFraction(f2);
  return bf1.multiply(bf2);
};

Future<BigFraction> future = mExecutorService.submit(task);
```

Overview of Interfaces Related to ExecutorService

- ExecutorService.submit() can initiate an async computation in Java.
- It returns a **`Future`** implemented as a **`FutureTask`**



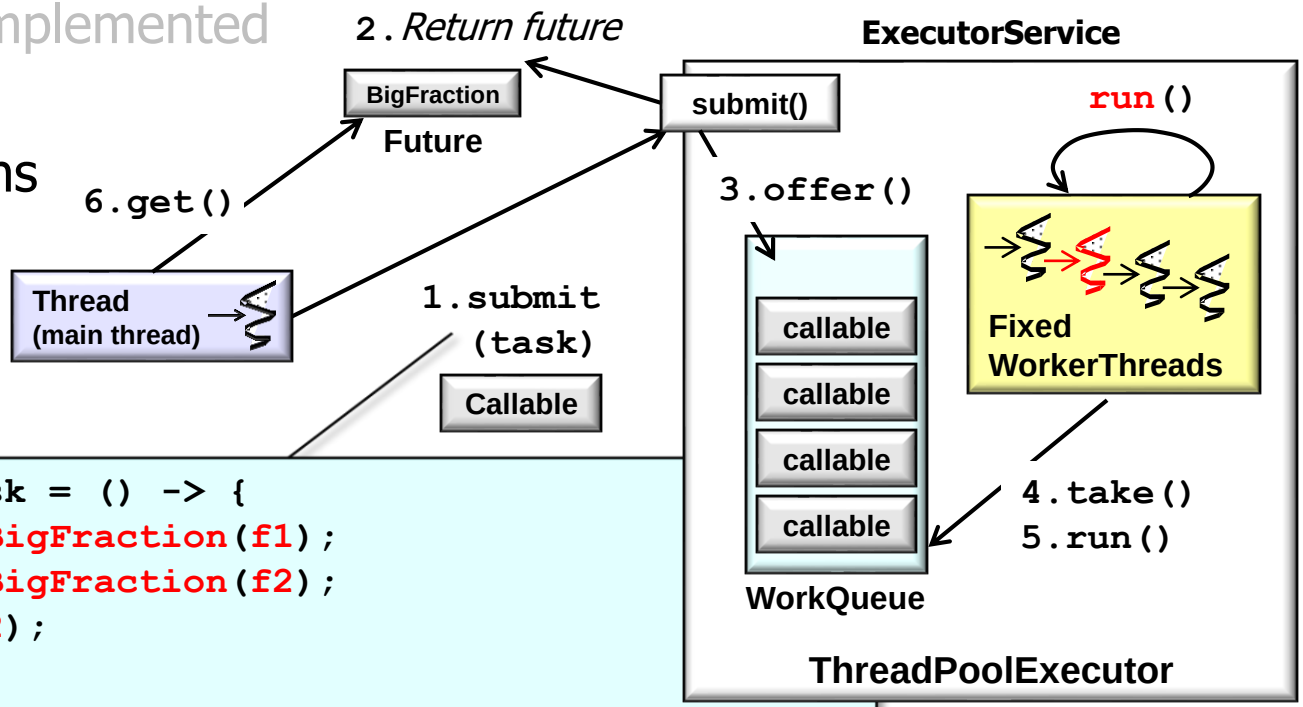

```
Callable<BigFraction> task = () -> {
  BigFraction bf1 = new BigFraction(f1);
  BigFraction bf2 = new BigFraction(f2);
  return bf1.multiply(bf2);
};

Future<BigFraction> future = mExecutorService.submit(task);
```

See upcoming lesson on "*Java FutureTask*"

Overview of Interfaces Related to ExecutorService

- ExecutorService.submit() can initiate an async computation in Java.
- It returns a `Future` implemented as a `FutureTask`
- Async computation runs in a worker thread



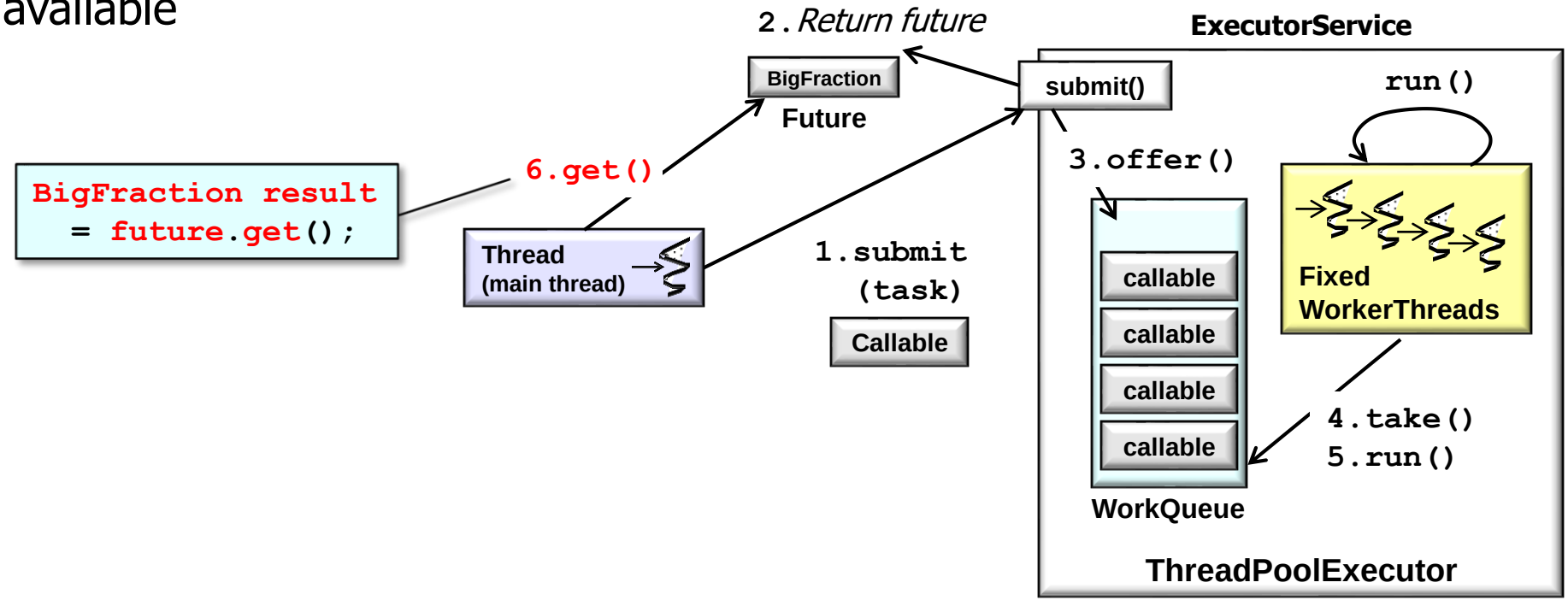

```
Callable<BigFraction> task = () -> {
  BigFraction bf1 = new BigFraction(f1);
  BigFraction bf2 = new BigFraction(f2);
  return bf1.multiply(bf2);
};

Future<BigFraction> future = mExecutorService.submit(task);
```

See upcoming part of this lesson on "*Overview of Java ThreadPoolExecutor*"

Overview of Interfaces Related to ExecutorService

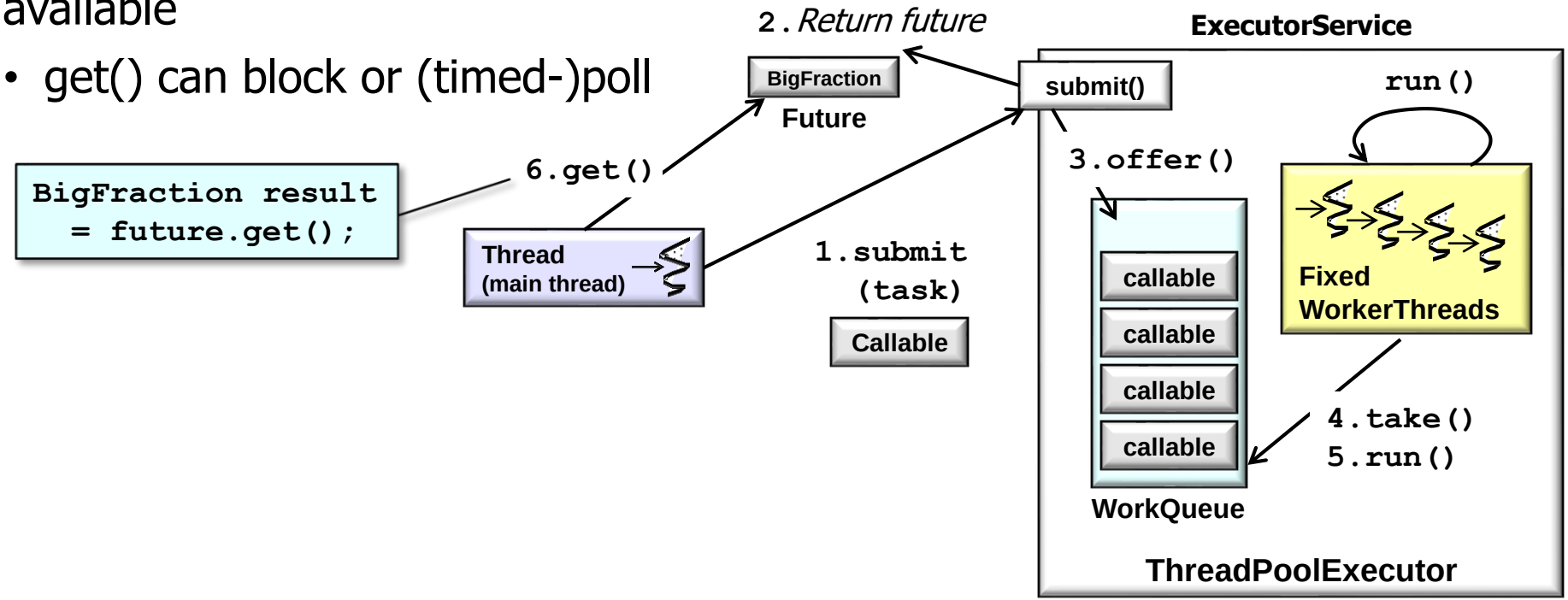
- When the async computation completes the future is triggered & the result is available



See www.nurkiewicz.com/2013/02/javautilconcurrentfuture-basics.html

Overview of Interfaces Related to ExecutorService

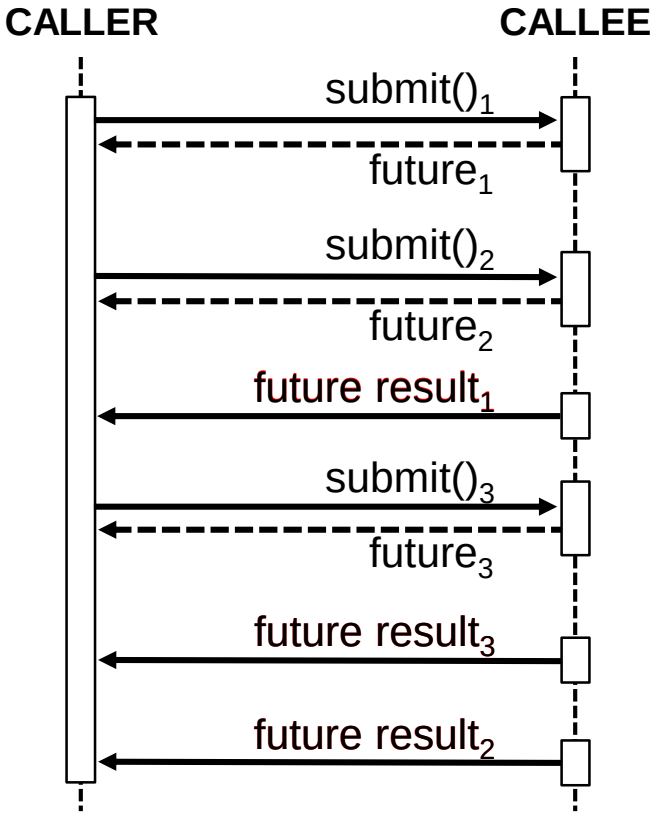
- When the async computation completes the future is triggered & the result is available
  - get() can block or (timed-)poll



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

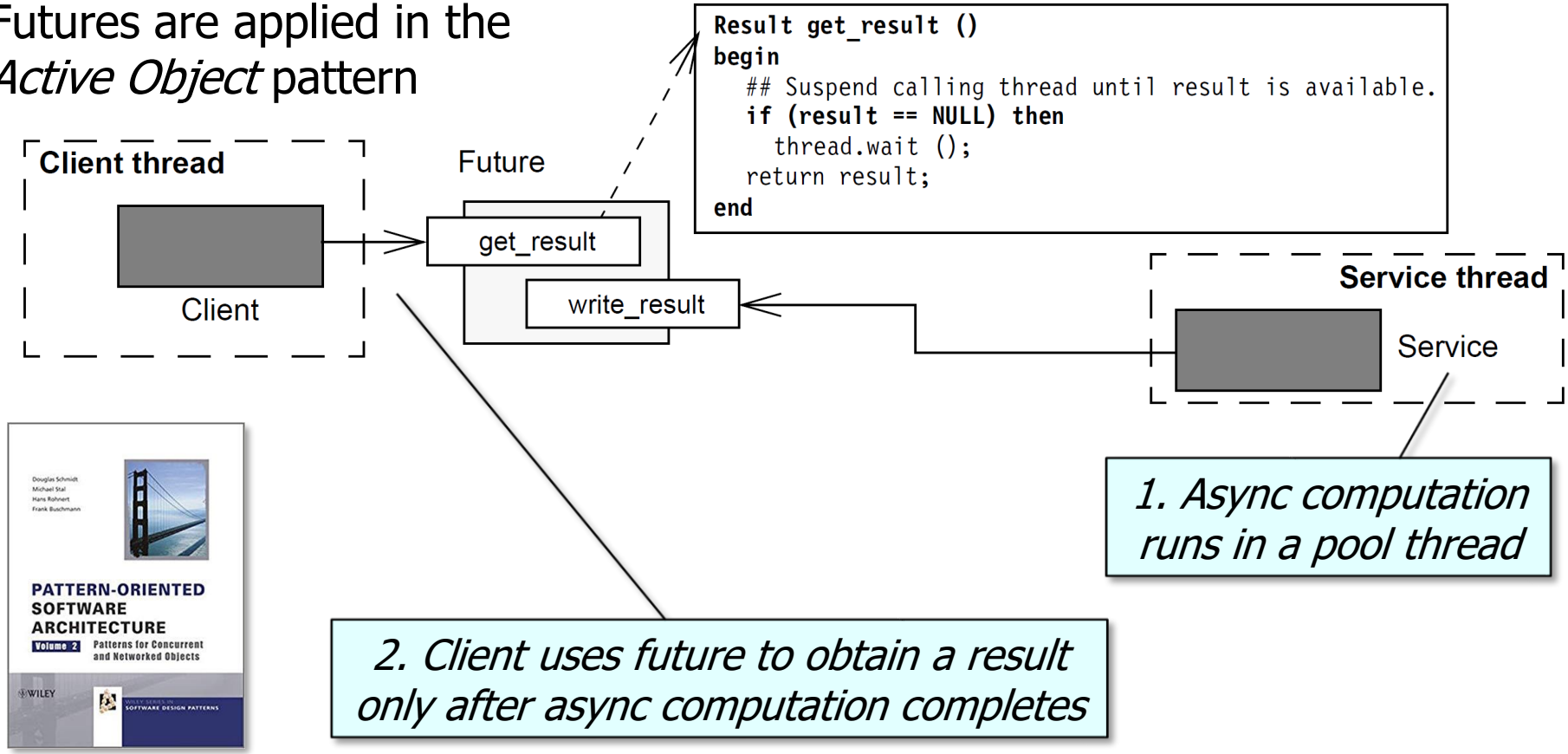
Overview of Interfaces Related to ExecutorService

- When the async computation completes the future is triggered & the result is available
  - get() can block or (timed-)poll
  - Results can occur in a different order than the original calls were made



Overview of Interfaces Related to ExecutorService

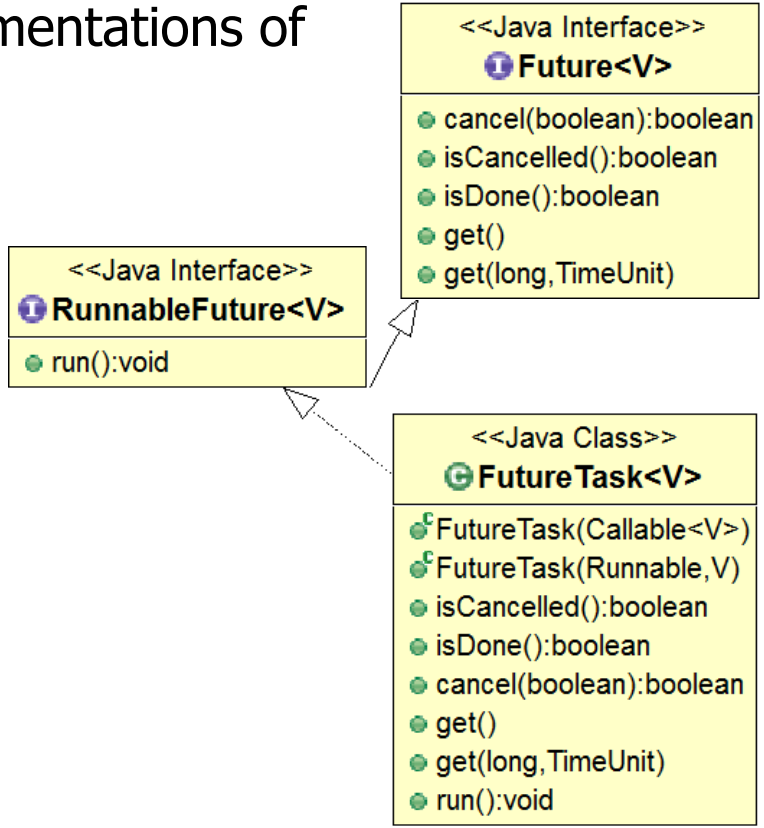
- Futures are applied in the *Active Object* pattern



See en.wikipedia.org/wiki/Active_object

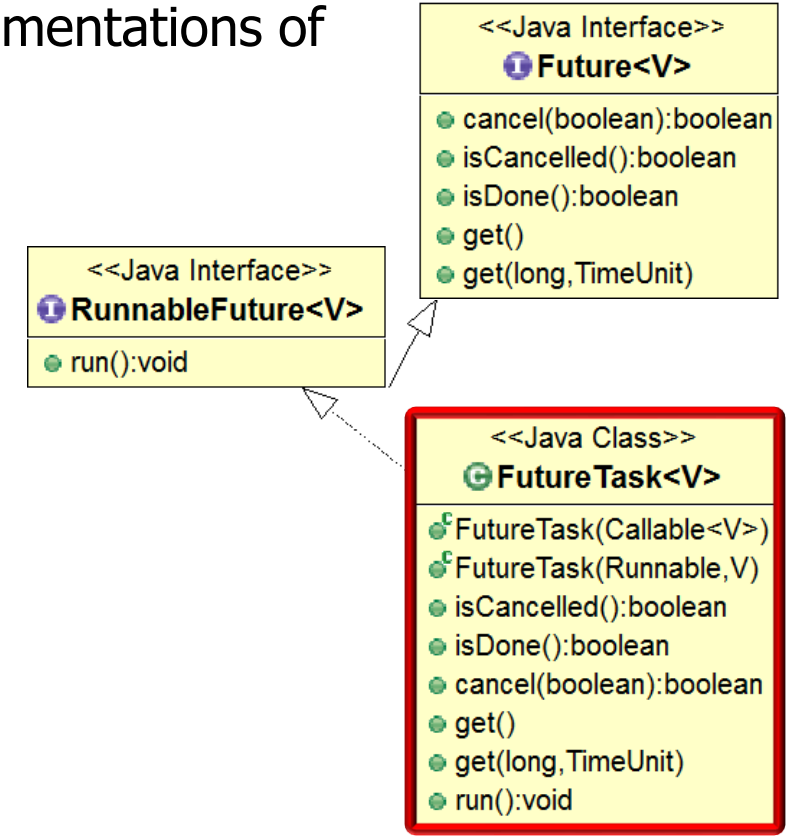
Overview of Interfaces Related to ExecutorService

- Other variants of Future are applied by implementations of the ExecutorService



Overview of Interfaces Related to ExecutorService

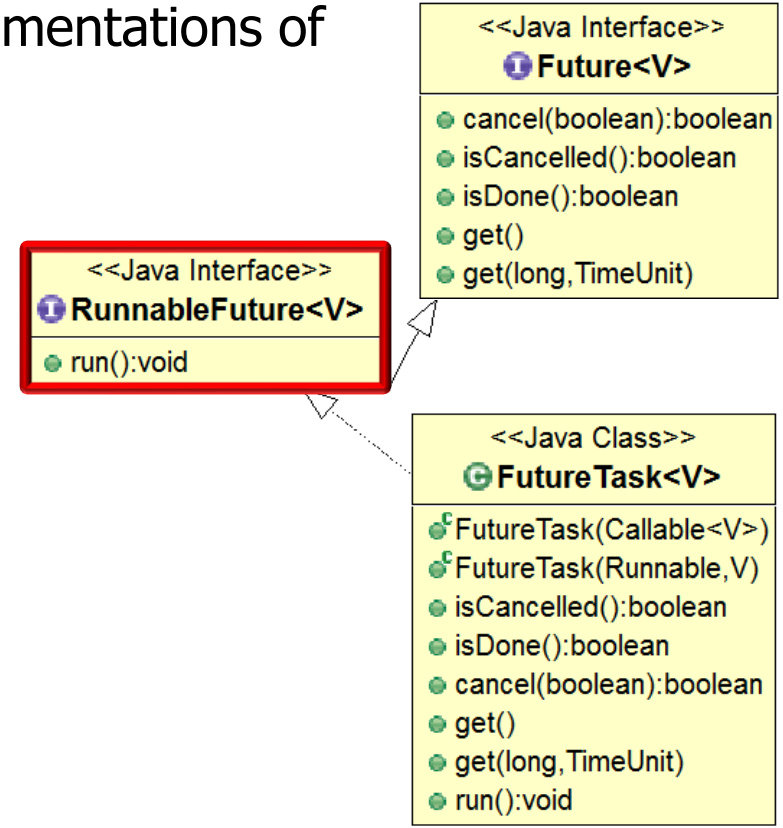
- Other variants of Future are applied by implementations of the ExecutorService, e.g.
  - **FutureTask**
    - Conveys result from thread running a task to thread(s) retrieving result



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

Overview of Interfaces Related to ExecutorService

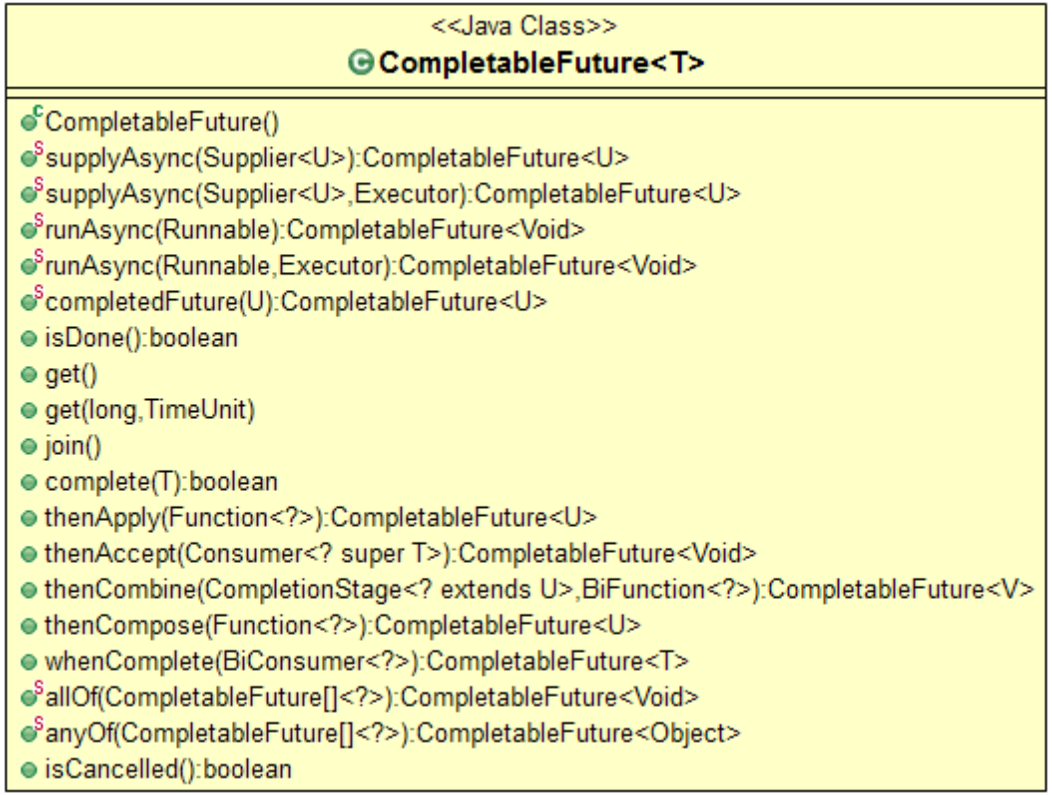
- Other variants of Future are applied by implementations of the ExecutorService, e.g.
  - FutureTask
  - **RunnableFuture**
    - Execution of run() method will complete the future & allow access to its results



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

Overview of Interfaces Related to ExecutorService

- A CompletableFuture is an implementation of Future that supports dependent actions triggered upon an async operation completion

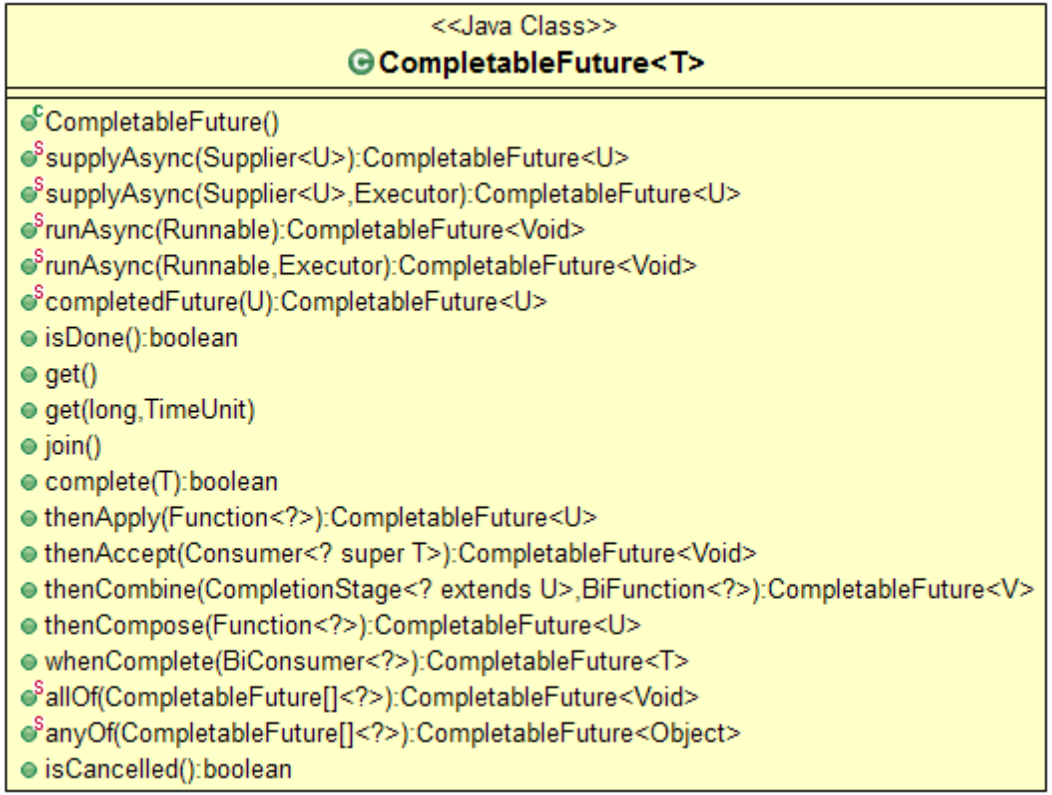


CompletableFuture isn't part of the Java Executor framework

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

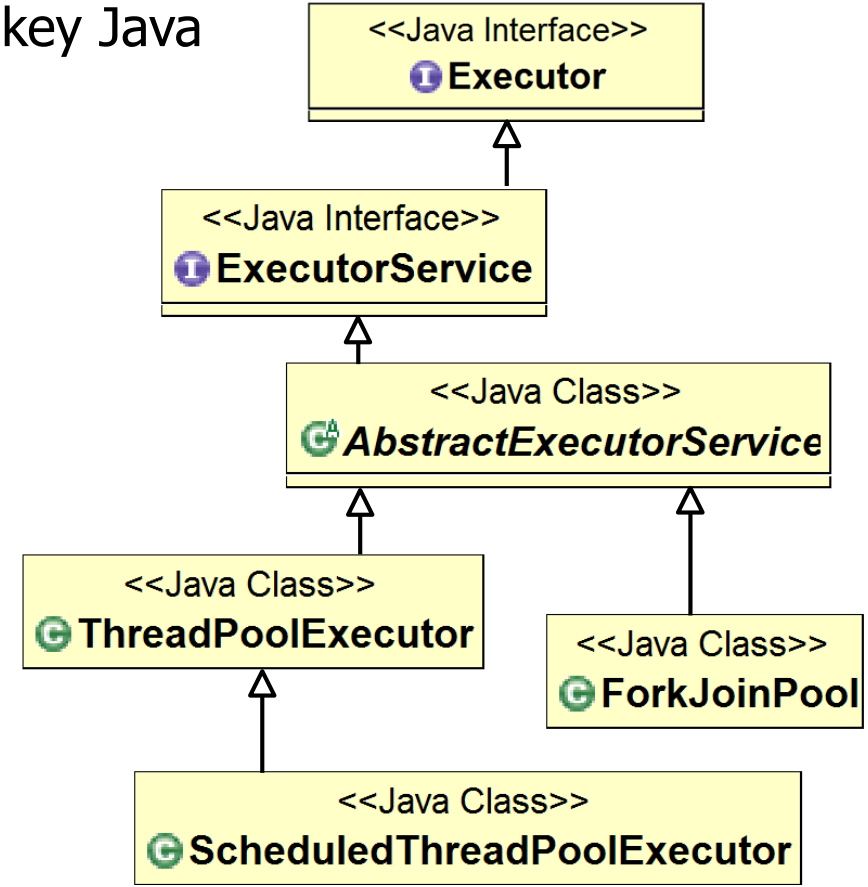
Overview of Interfaces Related to ExecutorService

- A CompletableFuture is an implementation of Future that supports dependent actions triggered upon an async operation completion
  - This topic is covered elsewhere



Overview of Interfaces Related to ExecutorService

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



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

End of Java Executor Service: Related Interfaces