Key Methods in the Java ExecutorService (Part 1)

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

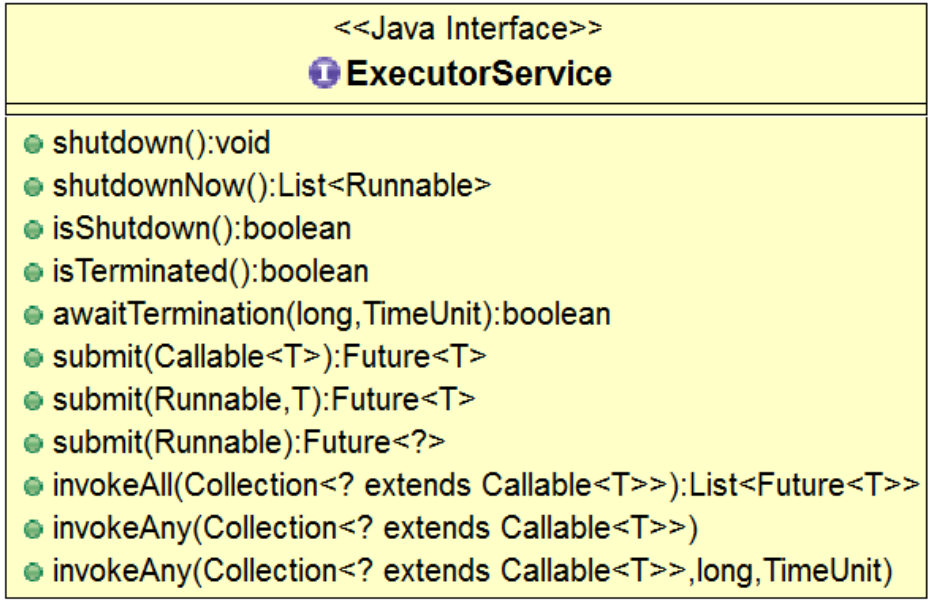
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

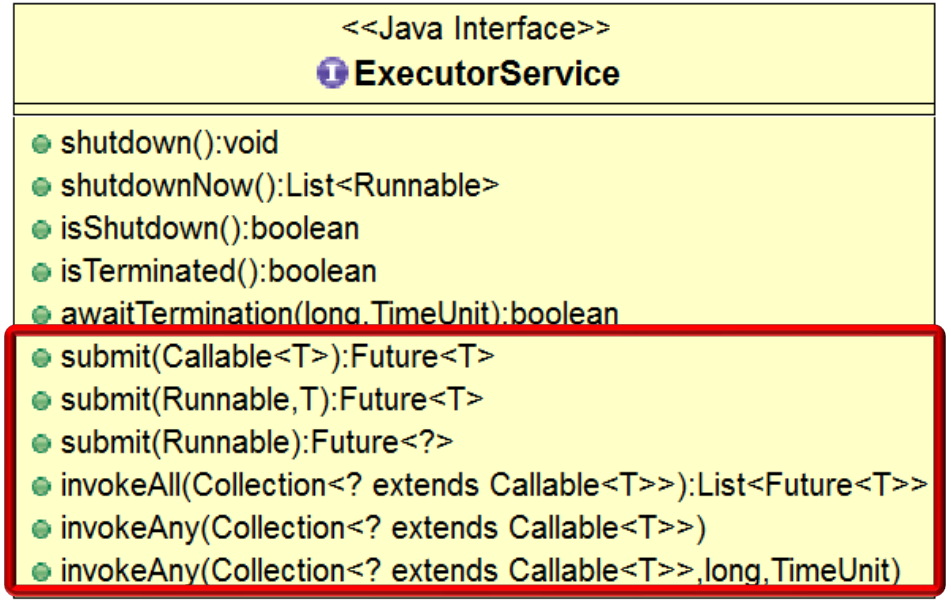
Learning Objectives in this Part of the Lesson

- Recognize the powerful features defined in the Java ExecutorService interface
- Understand other interfaces related to ExecutorService
- Know the key methods provided by ExecutorService



Learning Objectives in this Part of the Lesson

- Recognize the powerful features defined in the Java ExecutorService interface
- Understand other interfaces related to ExecutorService
- Know the key methods provided by ExecutorService
  - These methods submit 1+ tasks for asynchronous execution



Key Methods in the ExecutorService Interface: Task Execution

Key Methods in the ExecutorService Interface

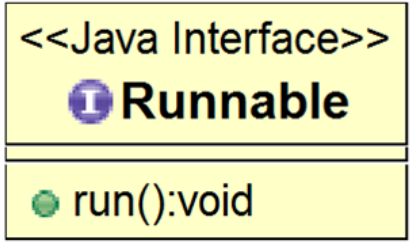
- ExecutorService executes tasks

```
public interface ExecutorService
                    extends Executor {
  // Inherited from Executor
  void execute(Runnable command);

  <T> Future<T> submit
          (Callable<T> task);
  ...
```

Key Methods in the ExecutorService Interface

- ExecutorService executes tasks
  - execute() runs one-way tasks whose run() hook method returns void


```
public interface ExecutorService
                    extends Executor {
  // Inherited from Executor
  void execute(Runnable command);

  <T> Future<T> submit
          (Callable<T> task);
  ...
```

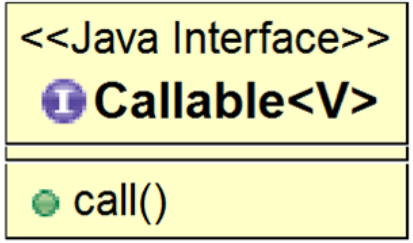


However, this method isn't very useful/common in practice

Key Methods in the ExecutorService Interface

- ExecutorService executes tasks
  - execute() runs one-way tasks whose run() hook method returns void
  - submit() runs two-way async tasks whose call() hook method returns a value


```
public interface ExecutorService
                    extends Executor {
  // Inherited from Executor
  void execute(Runnable command);

  <T> Future<T> submit
          (Callable<T> task);
  ...
```

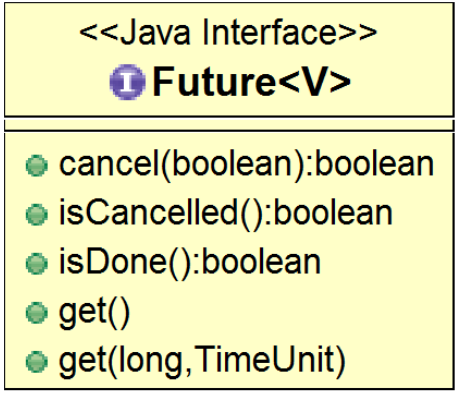


This method is the most useful/common in practice

Key Methods in the ExecutorService Interface

- ExecutorService executes tasks
  - execute() runs one-way tasks whose run() hook method returns void
  - submit() runs two-way async tasks whose call() hook method returns a value
    - This value can be obtained via a future

```
public interface ExecutorService
                    extends Executor {
  // Inherited from Executor
  void execute(Runnable command);

  <T> Future<T> submit
          (Callable<T> task);
  ...
```



Key Methods in the ExecutorService Interface

- ExecutorService executes tasks
  - execute() runs one-way tasks whose run() hook method returns void
  - submit() runs two-way async tasks whose call() hook method returns a value
    - This value can be obtained via a future
      - Supports the "synchronous future" processing model

```
public interface ExecutorService
                    extends Executor {
  // Inherited from Executor
  void execute(Runnable command);

  <T> Future<T> submit
          (Callable<T> task);
  ...
```

Key Methods in the ExecutorService Interface

- ExecutorService executes tasks
  - execute() runs one-way tasks whose run() hook method returns void
  - submit() runs two-way async tasks whose call() hook method returns a value
    - This value can be obtained via a future
    - Future.get() can block until task completes successfully

```
public interface ExecutorService
                    extends Executor {
  // Inherited from Executor
  void execute(Runnable command);

  <T> Future<T> submit
          (Callable<T> task);
  ...
```

Key Methods in the ExecutorService Interface

- ExecutorService executes tasks
  - execute() runs one-way tasks whose run() hook method returns void
  - submit() runs two-way async tasks whose call() hook method returns a value
    - This value can be obtained via a future
    - Future.get() can block until task completes successfully
      - After which point get() returns the task's result

```
public interface ExecutorService
                    extends Executor {
  // Inherited from Executor
  void execute(Runnable command);

  <T> Future<T> submit
          (Callable<T> task);
  ...
```

Key Methods in the ExecutorService Interface

- ExecutorService executes tasks
  - execute() runs one-way tasks whose run() hook method returns void
  - submit() runs two-way async tasks whose call() hook method returns a value
  - submit() can also run one-way async tasks that return no value

```
public interface ExecutorService
                    extends Executor {
  // Inherited from Executor
  void execute(Runnable command);

  <T> Future<T> submit
          (Callable<T> task);

  <T> Future<T> submit
          (Runnable task);
  ...
```

Key Methods in the ExecutorService Interface

- ExecutorService executes tasks
  - execute() runs one-way tasks whose run() hook method returns void
  - submit() runs two-way async tasks whose call() hook method returns a value
  - submit() can also run one-way async tasks that return no value
    - It is possible to cancel this computation, however

```
public interface ExecutorService
                    extends Executor {
  // Inherited from Executor
  void execute(Runnable command);

  <T> Future<T> submit
          (Callable<T> task);

  <T> Future<T> submit
          (Runnable task);
  ...
```

Key Methods in the ExecutorService Interface

- ExecutorService can also execute groups of tasks

```
public interface ExecutorService
                    extends Executor {
    ...
    <T> List<Future<T>> invokeAll
        (Collection<? extends
         Callable<T>> tasks) ...;

    <T> T invokeAny
        (Collection<? extends
         Callable<T>> tasks) ...;
    ...
```

Key Methods in the ExecutorService Interface

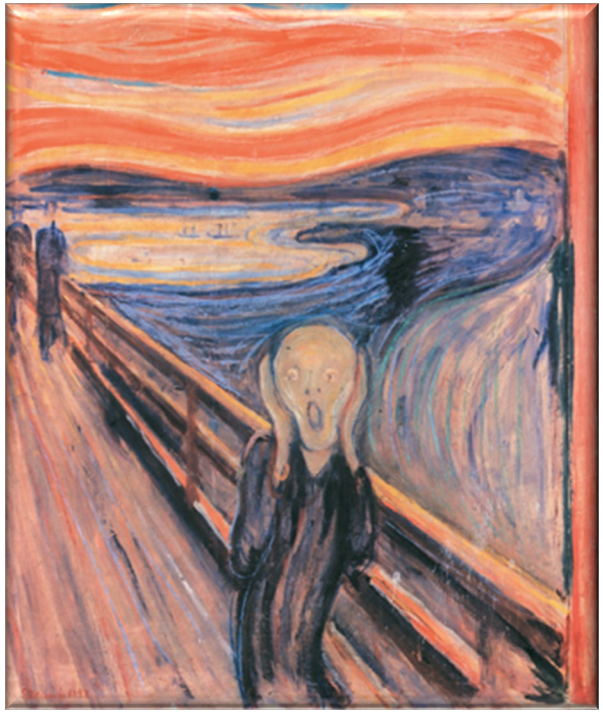
- ExecutorService can also execute groups of tasks

```
public interface ExecutorService
                    extends Executor {
  ...
  <T> List<Future<T>> invokeAll
      (Collection<? extends
       Callable<T>> tasks) ...;

  <T> T invokeAny
      (Collection<? extends
       Callable<T>> tasks) ...;
  ...
```

Groups of tasks can be passed to these methods as collection parameters

Key Methods in the ExecutorService Interface

- ExecutorService can also execute groups of tasks



```
public interface ExecutorService
                    extends Executor {
  ...
  <T> List<Future<T>> invokeAll
      (Collection<? extends
       Callable<T>> tasks) ...;

  <T> T invokeAny
      (Collection<? extends
       Callable<T>> tasks) ...;
  ...
```

Don’t modify collection param while invokeAll() or invokeAny() are running!!!

Key Methods in the ExecutorService Interface

- ExecutorService can also execute groups of tasks
  - Returns a list of futures when all tasks complete

```
public interface ExecutorService
                 extends Executor {
  ...
 <T> List<Future<T>> invokeAll
     (Collection<? extends
      Callable<T>> tasks) ...;

 <T> T invokeAny
     (Collection<? extends
      Callable<T>> tasks) ...;
 ...
```

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

Key Methods in the ExecutorService Interface

- ExecutorService can also execute groups of tasks
  - Returns a list of futures when all tasks complete

```
public interface ExecutorService
                    extends Executor {
  ...
  <T> List<Future<T>> invokeAll
      (Collection<? extends
       Callable<T>> tasks) ...;

  <T> T invokeAny
      (Collection<? extends
       Callable<T>> tasks) ...;
  ...
```

All futures returned in this list are "done"!

Futures are used to indicate whether task terminate normally or exceptionally

Key Methods in the ExecutorService Interface

- ExecutorService can also execute groups of tasks
  - Returns a list of futures when all tasks complete
  - Return the result of *one* successful completion


```
public interface ExecutorService
                    extends Executor {
  ...
  <T> List<Future<T>> invokeAll
      (Collection<? extends
       Callable<T>> tasks) ...;

  <T> T invokeAny
      (Collection<? extends
       Callable<T>> tasks) ...;
  ...
```

Useful for concurrent algorithms that just want the result that completes first

Key Methods in the ExecutorService Interface

- ExecutorService can also execute groups of tasks
  - Returns a list of futures when all tasks complete
  - Return the result of *one* successful completion
    - Cancel uncompleted tasks


```
public interface ExecutorService
                    extends Executor {
  ...
  <T> List<Future<T>> invokeAll
      (Collection<? extends
       Callable<T>> tasks) ...;

  <T> T invokeAny
      (Collection<? extends
       Callable<T>> tasks) ...;
  ...
```

Key Methods in the ExecutorService Interface

- ExecutorService can also execute groups of tasks
  - Returns a list of futures when all tasks complete
  - Return the result of *one* successful completion
    - Cancel uncompleted tasks
    - Ignore other completed task results


```
public interface ExecutorService
                    extends Executor {
  ...
  <T> List<Future<T>> invokeAll
      (Collection<? extends
       Callable<T>> tasks) ...;

  <T> T invokeAny
      (Collection<? extends
       Callable<T>> tasks) ...;
  ...
```

Key Methods in the ExecutorService Interface

- ExecutorService can also execute groups of tasks
  - Returns a list of futures when all tasks complete
  - Return the result of *one* successful completion

```
public interface ExecutorService
                    extends Executor {
  ...
  <T> List<Future<T>> invokeAll
      (Collection<? extends
       Callable<T>> tasks) ...;

  <T> T invokeAny
      (Collection<? extends
       Callable<T>> tasks) ...;
  ...
```

These methods block the calling thread until they are finished, which may be non-intuitive..

Key Methods in the ExecutorService Interface

- ExecutorService can also execute groups of tasks
  - Returns a list of futures when all tasks complete
  - Return the result of *one* successful completion

```
public interface ExecutorService
                    extends Executor {
  ...
  <T> List<Future<T>> invokeAll
      (Collection<? extends
       Callable<T>> tasks,
       long timeout, TimeUnit unit)
       ...;

  <T> T invokeAny(Collection<?
       extends Callable<T>> tasks,
       long timeout, TimeUnit unit)
       ...;
  ...
```

These overloaded methods block for up to a given amount of time

Key Methods in the ExecutorService Interface

- ExecutorService can also execute groups of tasks
  - Returns a list of futures when all tasks complete
  - Return the result of *one* successful completion

```
public interface ExecutorService
                    extends Executor {
  ...
  <T> List<Future<T>> invokeAll
      (Collection<? extends
       Callable<T>> tasks,
       long timeout, TimeUnit unit)
       ...;

  <T> T invokeAny(Collection<?
       extends Callable<T>> tasks,
       long timeout, TimeUnit unit)
       ...;
  ...
```

If method didn't time out, each task completed, whereas if it did time out, some tasks will not have completed.

Task that have not completed are cancelled if timeout occurs.

Key Methods in the ExecutorService Interface

- ExecutorService can also execute groups of tasks
  - Returns a list of futures when all tasks complete
  - Return the result of *one* successful completion

```
public interface ExecutorService
                    extends Executor {
  ...
  <T> List<Future<T>> invokeAll
      (Collection<? extends
       Callable<T>> tasks,
       long timeout, TimeUnit unit)
       ...;

  <T> T invokeAny(Collection<?
       extends Callable<T>> tasks,
       long timeout, TimeUnit unit)
       ...;
  ...
```

TimeoutException is thrown if timeout elapses

End of Key Methods in the Java ExecutorService (Part 1)