Java Executor: Introduction

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 single simple feature provided by the Java Executor interface

**Interface Executor**

All Known Subinterfaces:
- ExecutorService, ScheduledExecutorService

All Known Implementing Classes:
- AbstractExecutorService, ForkJoinPool, ScheduledThreadPoolExecutor, ThreadPoolExecutor

```java
public interface Executor

An object that executes submitted Runnable tasks. This interface provides a way of decoupling task submission from the mechanics of how each task will be run, including details of thread use, scheduling, etc. An Executor is normally used instead of explicitly creating threads. For example, rather than invoking new Thread(new RunnableTask()).start() for each of a set of tasks, you might use:

Executor executor = anExecutor;
executor.execute(new RunnableTask1());
executor.execute(new RunnableTask2());
...
```

However, the Executor interface does not strictly require that execution be asynchronous. In the simplest case, an executor can run the submitted task immediately in the caller's thread:
Overview of the Java Executor Interface
Overview of the Java Executor Interface

- Provides a method to submit new tasks for execution

```java
<<Java Interface>>
Executor

execute(Runnable): void
```

Defines a simple functional interface that decouples task submission from the mechanics of how each task is run.

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executor.html](docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executor.html)
Overview of the Java Executor Interface

• Provides a method to submit new tasks for execution
• Each task implements the Runnable interface

Runnable is also a functional interface

See docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html
Provides a method to submit new tasks for execution
• Each task implements the Runnable interface
• Represents a “command” to execute

Command pattern

A command encapsulates all info needed to perform an action or trigger an event later in time

See en.wikipedia.org/wiki/Command_pattern
Overview of the Java Executor Interface

- Provides a method to submit new tasks for execution
- Each task implements the Runnable interface
  - Represents a “command” to execute
- Can also implement Command Processor pattern

Packages a piece of application functionality—as well as its parameterization in an object—to make it usable in another context

See [www.dre.vanderbilt.edu/~schmidt/CommandProcessor.pdf](http://www.dre.vanderbilt.edu/~schmidt/CommandProcessor.pdf)
Overview of the Java Executor Interface

- Provides a method to submit new tasks for execution
- Each task implements the Runnable interface
  - Represents a “command” to execute
  - Can also implement Command Processor pattern
- Provides “one-way” task semantics
  - i.e., run() computations return no results

Results are typically “returned” via side-effects
Overview of the Java Executor Interface

- Provides a method to submit new tasks for execution
- Each task implements the Runnable interface
  - Represents a “command” to execute
  - Can also implement Command Processor pattern
  - Provides “one-way” task semantics
- Can execute in a background thread or main thread
  - Depending on Executor interface’s implementation

There’s even a single-threaded executor implementation!
End of Java Executor: Introduction