

The Java Executor 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 single simple feature provided by the Java Executor interface



Interface Executor

All Known Subinterfaces:

`ExecutorService`, `ScheduledExecutorService`

All Known Implementing Classes:

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

```
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 Interface>>

I 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

Overview of the Java Executor Interface

- Provides a method to submit new tasks for execution

<<Java Interface>>

Executor

execute(Runnable):void

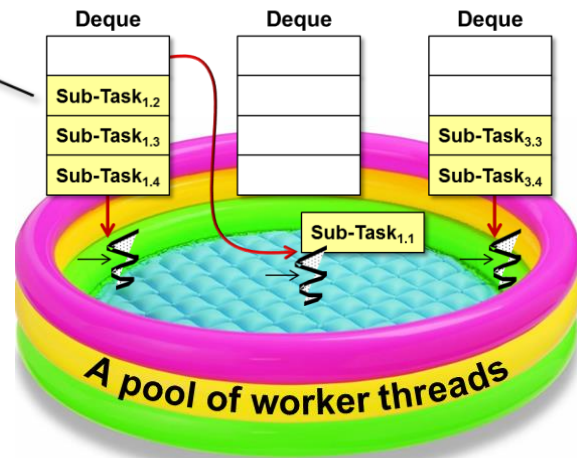
The Executor interface can be implemented via various thread pooling mechanisms



Fixed-sized Thread Pool



Variable-sized Thread Pool



Work-stealing Thread Pool

There's even a single threaded implementation of Executor!

Overview of the Java Executor Interface

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

<<Java Interface>>

 **Runnable**

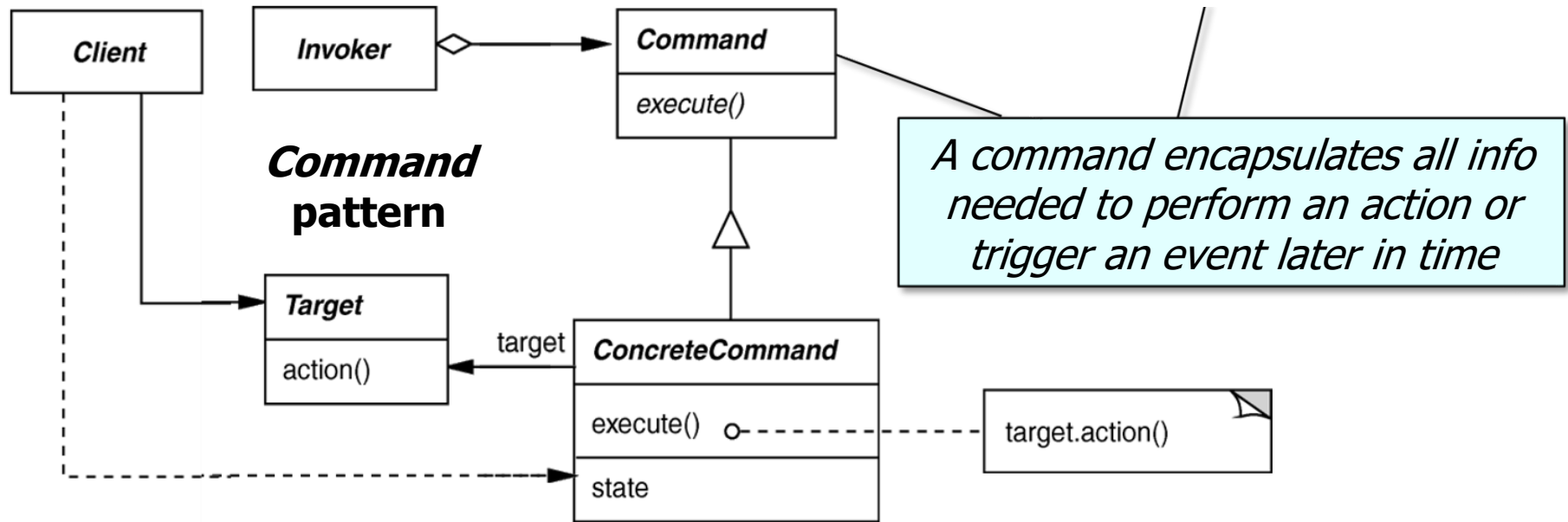
 run():void

Runnable is also a functional interface

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

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



See en.wikipedia.org/wiki/Command_pattern

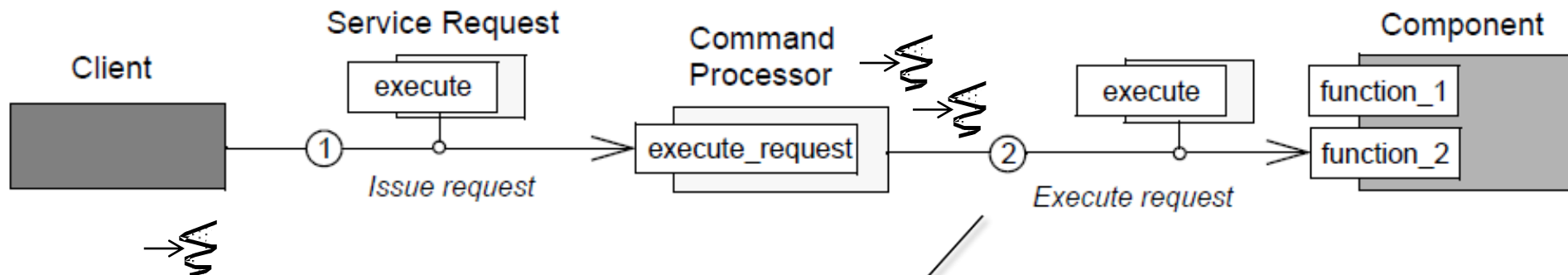
Overview of the Java Executor Interface

- 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

<<Java Interface>>

Runnable

run():void



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

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., does not return a result

<<Java Interface>>

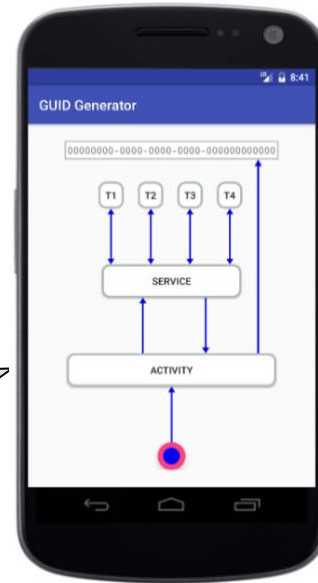
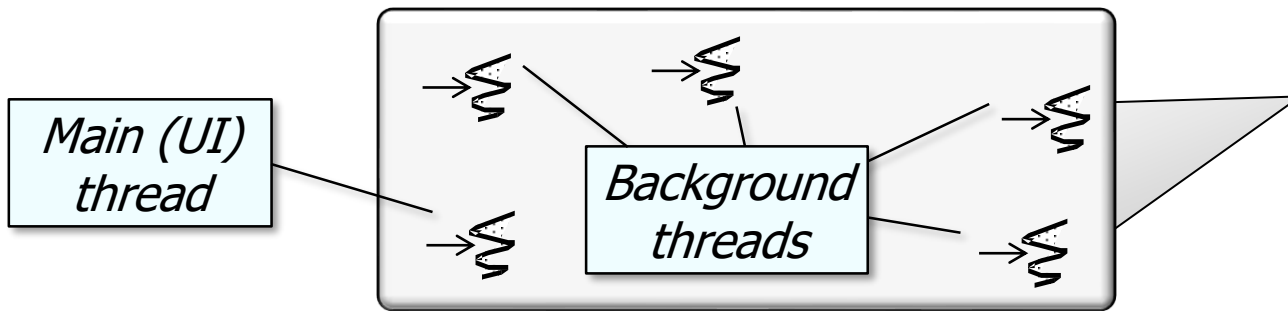
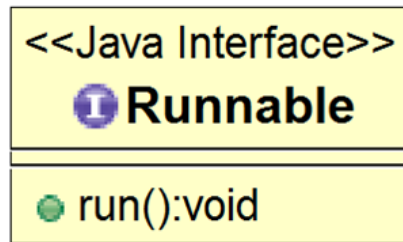
 **Runnable**

 run():void



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



End of Java Executor Interface (Part 1)