The Java Executor Framework:
The Java Executors Class

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

- Understand the purpose of the Java executor framework
- Recognize the features & benefits of thread pools
- Note a human known use of thread pools
- Know the Java Executor framework thread pools
- Learn the key interfaces the framework provides
- Appreciate the factory methods provided by the Java Executors class
The Java Executors Class
The Java Executors Class

- Executors is a utility class that creates executor implementations

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html
**The Java Executors Class**

- Executors is a utility class that creates executor implementations.
- A utility class is a final class having only static methods, no state, & a private constructor.

The Executors utility class has factory method that create desired executors

See en.wikipedia.org/wiki/Factory_method_pattern
The Executors utility class has a factory method that creates desired executors.

- Example factories include: cached, fixed, work-stealing thread pools, etc.
The Java Executors Class

- The Executors utility class also has a factory method that can be used to create new threads.

There's a default thread factory

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#defaultThreadFactory
The Java Executors Class

• The Executors utility class also has a factory method that can be used to create new threads

• The DefaultThreadFactory implements the Thread Factory interface

Interface ThreadFactory

public interface ThreadFactory

An object that creates new threads on demand. Using thread factories removes hardwiring of calls to new Thread, enabling applications to use special thread subclasses, priorities, etc.

The simplest implementation of this interface is just:

class SimpleThreadFactory implements ThreadFactory {
    public Thread newThread(Runnable r) {
        return new Thread(r);
    }
}

The Executors.defaultThreadFactory() method provides a more useful simple implementation, that sets the created thread context to known values before returning it.

Since:
1.5

Method Summary

<table>
<thead>
<tr>
<th>All Methods</th>
<th>Instance Methods</th>
<th>Abstract Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modifier and Type</td>
<td>Method and Description</td>
<td></td>
</tr>
<tr>
<td>Thread</td>
<td>newThread(Runnable r)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Constructs a new Thread.</td>
<td></td>
</tr>
</tbody>
</table>
The Executors utility class also has a factory method that can be used to create new threads.

- The DefaultThreadFactory implements the ThreadFactory interface.
- Many Executors factory methods use the default thread factory.

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#defaultThreadFactory
The Executors utility class also has a factory method that can be used to create new threads.

- The DefaultThreadFactory implements the Thread Factory interface.
- You can also define custom thread factories & pass them to factory methods.

End of the Java Executors Framework: The Java Executors Interface