Java ExecutorService:
Overview of Java ThreadPoolExecutor

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
- Be aware of how ThreadPoolExecutor implements ExecutorService
Overview of the Java ThreadPoolExecutor
Overview of the Java ThreadPoolExecutor

- ThreadPoolExecutor implements the ExecutorService interface
- Indirectly via the AbstractExecutorService super class
Overview of the Java ThreadPoolExecutor

- ThreadPoolExecutor runs each submitted task via a worker thread provided by a pool.

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ThreadPoolExecutor.html
Overview of the Java ThreadPoolExecutor

- ThreadPoolExecutor runs each submitted task via a worker thread provided by a pool.
Overview of the Java ThreadPoolExecutor

- ThreadPoolExecutor’s constructor can be configured via various parameters

```java
ThreadPoolExecutor
(int corePoolSize,
 int maximumPoolSize,
 long keepAliveTime,
 TimeUnit unit,
 BlockingQueue<Runnable> workQueue,
 ThreadFactory threadFactory)
```

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ThreadPoolExecutor.html#ThreadPoolExecutor
Overview of the Java ThreadPoolExecutor

- The # of threads in the pool can be controlled programmatically
- `corePoolSize` – # of threads to keep in the pool, even if they are idle
- `maximumPoolSize` – maximum # of threads to allow in the pool

```java
ThreadPoolExecutor
(int corePoolSize,
  int maximumPoolSize,
  long keepAliveTime,
  TimeUnit unit,
  BlockingQueue<Runnable>
  workQueue,
  ThreadFactory
  threadFactory)
```
Overview of the Java ThreadPoolExecutor

- The lifetime of threads in the pool can be controlled programmatically
  - `keepAliveTime` – maximum time that excess idle threads will wait for new tasks before terminating when # of threads is greater than the core
  - `unit` – the time unit for the `keepAliveTime` argument

```java
ThreadPoolExecutor
(int corePoolSize,
 int maximumPoolSize,
 long keepAliveTime,
 TimeUnit unit,
 BlockingQueue<Runnable>
 workQueue,
 ThreadFactory
 threadFactory)
```
The queue holding tasks submitted by the execute() & submit() methods can be controlled programmatically.

- **workQueue** – the queue to use for holding tasks before they are run.

```java
ThreadPoolExecutor
(int corePoolSize,
int maximumPoolSize,
long keepAliveTime,
TimeUnit unit,
BlockingQueue<Runnable>
workQueue,
ThreadFactory threadFactory)
```
Overview of the Java ThreadPoolExecutor

- The queue can be strategized
- Direct handoff (used by cached pool)

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/SynchronousQueue.html
Overview of the Java ThreadPoolExecutor

• The queue can be strategized
• Direct handoff (used by cached pool)
• Pros – Avoids deadlock when internal dependencies

```
ThreadPoolExecutor
(int corePoolSize,
 int maximumPoolSize,
 long keepAliveTime,
 TimeUnit unit,
 BlockingQueue<Runnable>
 workQueue,
 ThreadFactory
 threadFactory)
```

See asznajder.github.io/thread-pool-induced-deadlocks
Overview of the Java ThreadPoolExecutor

- The queue can be strategized
- Direct handoff (used by cached pool)
  - Pros – Avoids deadlock when internal dependencies
  - Cons – Can create unlimited threads

```
ThreadPoolExecutor
    (int corePoolSize,
     int maximumPoolSize,
     long keepAliveTime,
     TimeUnit unit,
     BlockingQueue<Runnable>
         workQueue,
     ThreadFactory
         threadFactory)
```
Overview of the Java ThreadPoolExecutor

- The queue can be strategized
  - Direct handoff
- Unbounded queues (used by fixed pool)

```java
ThreadPoolExecutor
    (int corePoolSize,
     int maximumPoolSize,
     long keepAliveTime,
     TimeUnit unit,
     BlockingQueue<Runnable>
        workQueue,
     ThreadFactory
        threadFactory)
```

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/LinkedBlockingQueue.html
Overview of the Java ThreadPoolExecutor

- The queue can be strategized
  - Direct handoff
- Unbounded queues (used by fixed pool)
  - Pros – Smooths bursty requests

```java
ThreadPoolExecutor
(int corePoolSize,
 int maximumPoolSize,
 long keepAliveTime,
 TimeUnit unit,
 BlockingQueue<Runnable> workQueue,
 ThreadFactory threadFactory)
```
Overview of the Java ThreadPoolExecutor

- The queue can be strategized
  - Direct handoff
- Unbounded queues (used by fixed pool)
  - Pros – Smooths bursty requests
  - Cons – Can consume unlimited resources

```
ThreadPoolExecutor
    (int corePoolSize,
     int maximumPoolSize,
     long keepAliveTime,
     TimeUnit unit,
     BlockingQueue<Runnable>
     workQueue,
     ThreadFactory
     threadFactory)
```
Overview of the Java ThreadPoolExecutor

• The queue can be strategized
  • Direct handoff
  • Unbounded queues
  • Bounded queues (also used by fixed pool)

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ArrayBlockingQueue.html
Overview of the Java ThreadPoolExecutor

- The queue can be strategized
  - Direct handoff
  - Unbounded queues
  - Bounded queues (also used by fixed pool)
- Pros – Limits resource utilization

ThreadPoolExecutor
(int corePoolSize,
int maximumPoolSize,
long keepAliveTime,
TimeUnit unit,
BlockingQueue<Runnable> workQueue,
ThreadFactory threadFactory)
Overview of the Java ThreadPoolExecutor

- The queue can be strategized
  - Direct handoff
  - Unbounded queues
  - Bounded queues (also used by fixed pool)
- Pros – Limits resource utilization
- Cons – Hard to tune & may deadlock

See asznajder.github.io/thread-pool-induced-deadlocks
Overview of the Java ThreadPoolExecutor

- The factory used to create threads can be controlled programmatically
- `threadFactory` – the factory to use when creating a new thread

```java
ThreadPoolExecutor
(int corePoolSize,
 int maximumPoolSize,
 long keepAliveTime,
 TimeUnit unit,
 BlockingQueue<Runnable>
 workQueue,
 ThreadFactory
 threadFactory)
```

ThreadFactory removes hardwiring of calls to new Thread, enabling programs to use special thread subclasses, priorities, etc.
End of JavaExecutor Service: Overview of Java ThreadPoolExecutor