Java ExecutorService: Overview of Java ThreadPoolExecutor

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

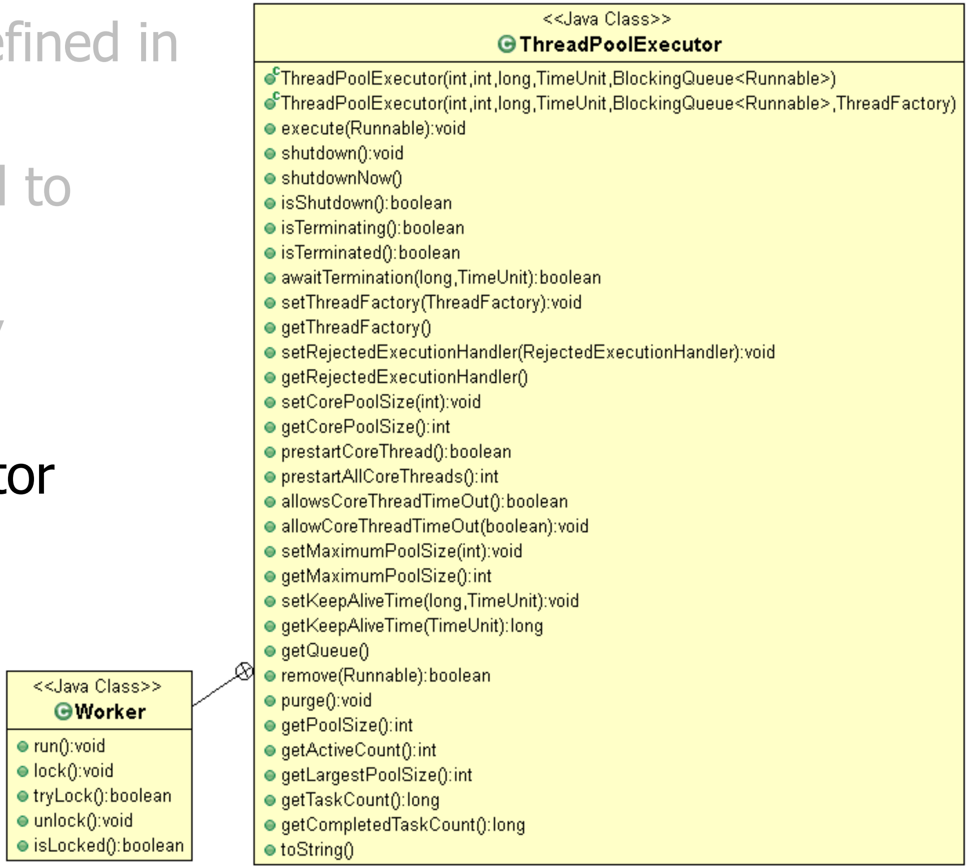
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

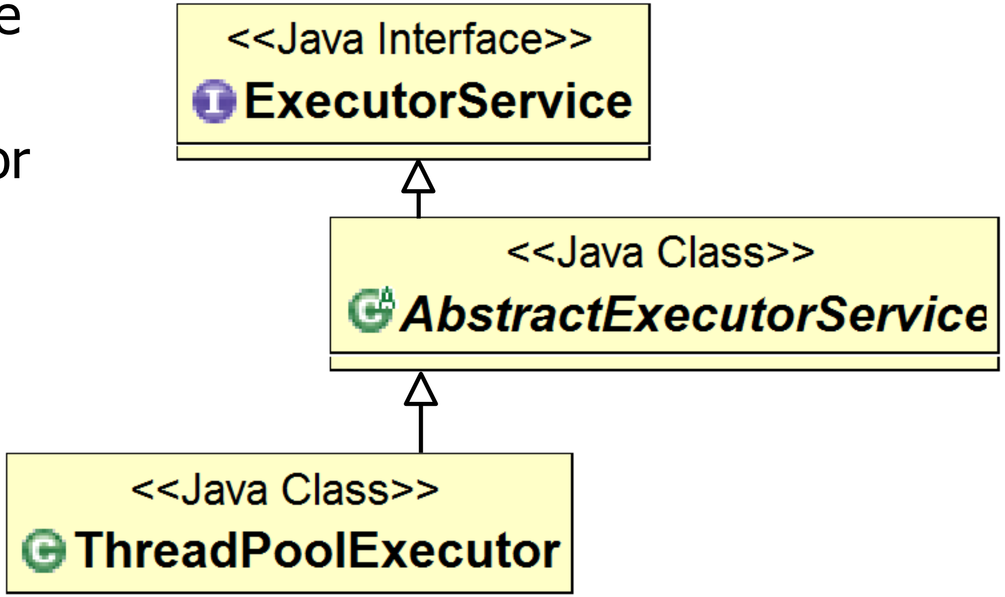
- Recognize the powerful features defined in the Java ExecutorService interface
- Understand other interfaces related to ExecutorService
- Know the key methods provided by ExecutorService
- Be aware of how ThreadPoolExecutor implements ExecutorService


<<Java Class>>
ThreadPoolExecutor
ThreadPoolExecutor(int,int,long,TimeUnit,BlockingQueue<Runnable>)
ThreadPoolExecutor(int,int,long,TimeUnit,BlockingQueue<Runnable>,ThreadFactory)
execute(Runnable):void
shutdown():void
shutdownNow()
isShutdown():boolean
isTerminating():boolean
isTerminated():boolean
awaitTermination(long,TimeUnit):boolean
setThreadFactory(ThreadFactory):void
getThreadFactory()
setRejectedExecutionHandler(RejectedExecutionHandler):void
getRejectedExecutionHandler()
setCorePoolSize(int):void
getCorePoolSize():int
prestartCoreThread():boolean
prestartAllCoreThreads():int
allowsCoreThreadTimeOut():boolean
allowCoreThreadTimeOut(boolean):void
setMaximumPoolSize(int):void
getMaximumPoolSize():int
setKeepAliveTime(long,TimeUnit):void
getKeepAliveTime(TimeUnit):long
getQueue()
remove(Runnable):boolean
purge():void
getPoolSize():int
getActiveCount():int
getLargestPoolSize():int
getTaskCount():long
getCompletedTaskCount():long
toString()
<<Java Class>>
Worker
run():void
lock():void
tryLock():boolean
unlock():void
isLocked():boolean

Overview of the Java ThreadPoolExecutor

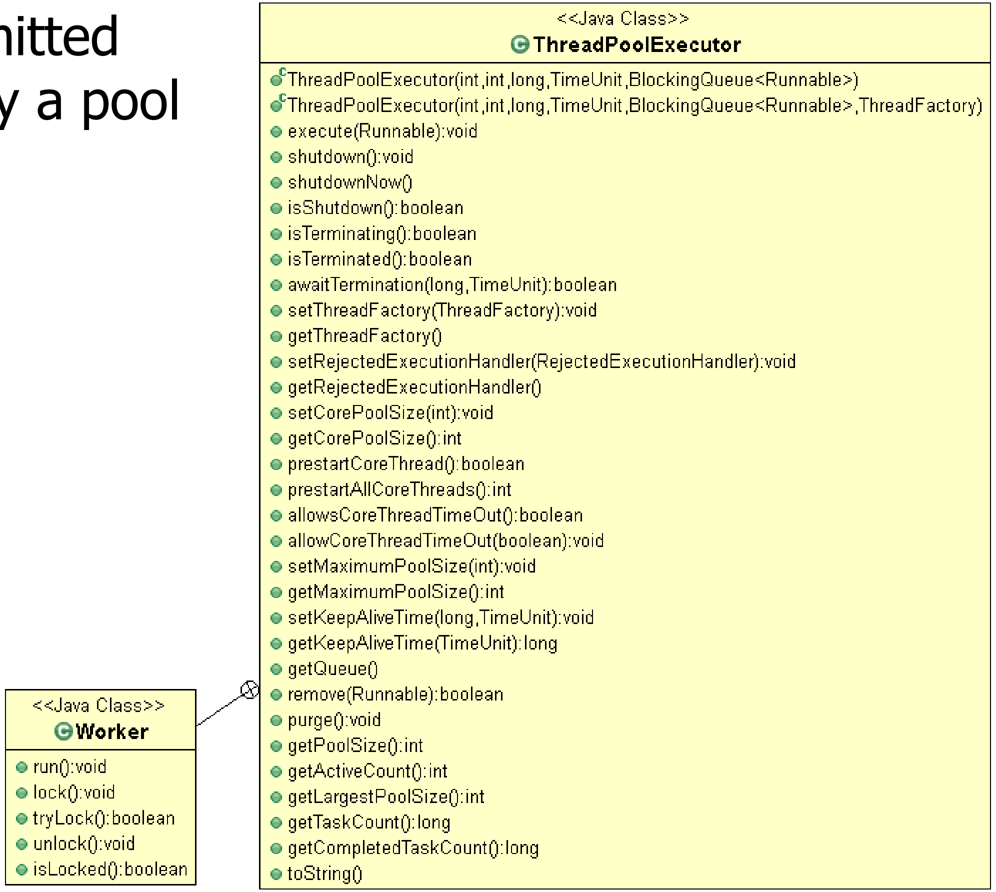
Overview of the Java ThreadPoolExecutor

- ThreadPoolExecutor implements the ExecutorService interface
  - Indirectly via the AbstractExecutor Service super class



Overview of the Java ThreadPoolExecutor

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

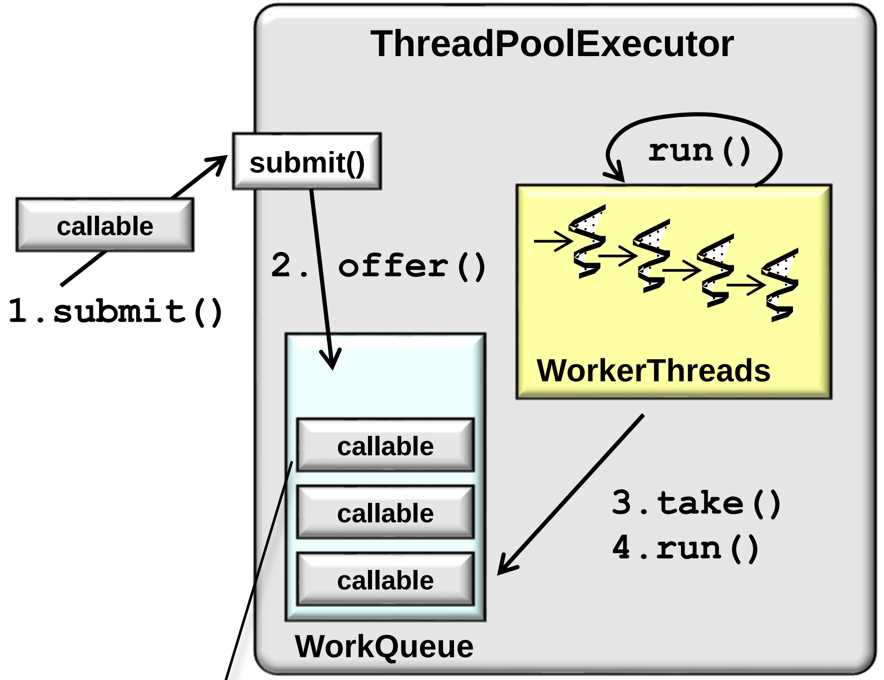


<<Java Class>>
ThreadPoolExecutor
ThreadPoolExecutor(int,int,long,TimeUnit,BlockingQueue<Runnable>)
ThreadPoolExecutor(int,int,long,TimeUnit,BlockingQueue<Runnable>,ThreadFactory)
execute(Runnable):void
shutdown():void
shutdownNow()
isShutdown():boolean
isTerminating():boolean
isTerminated():boolean
awaitTermination(long,TimeUnit):boolean
setThreadFactory(ThreadFactory):void
getThreadFactory()
setRejectedExecutionHandler(RejectedExecutionHandler):void
getRejectedExecutionHandler()
setCorePoolSize(int):void
getCorePoolSize():int
prestartCoreThread():boolean
prestartAllCoreThreads():int
allowsCoreThreadTimeOut():boolean
allowCoreThreadTimeOut(boolean):void
setMaximumPoolSize(int):void
getMaximumPoolSize():int
setKeepAliveTime(long,TimeUnit):void
getKeepAliveTime(TimeUnit):long
getQueue()
remove(Runnable):boolean
purge():void
getPoolSize():int
getActiveCount():int
getLargestPoolSize():int
getTaskCount():long
getCompletedTaskCount():long
toString()
<<Java Class>>
Worker
run():void
lock():void
tryLock():boolean
unlock():void
isLocked():boolean

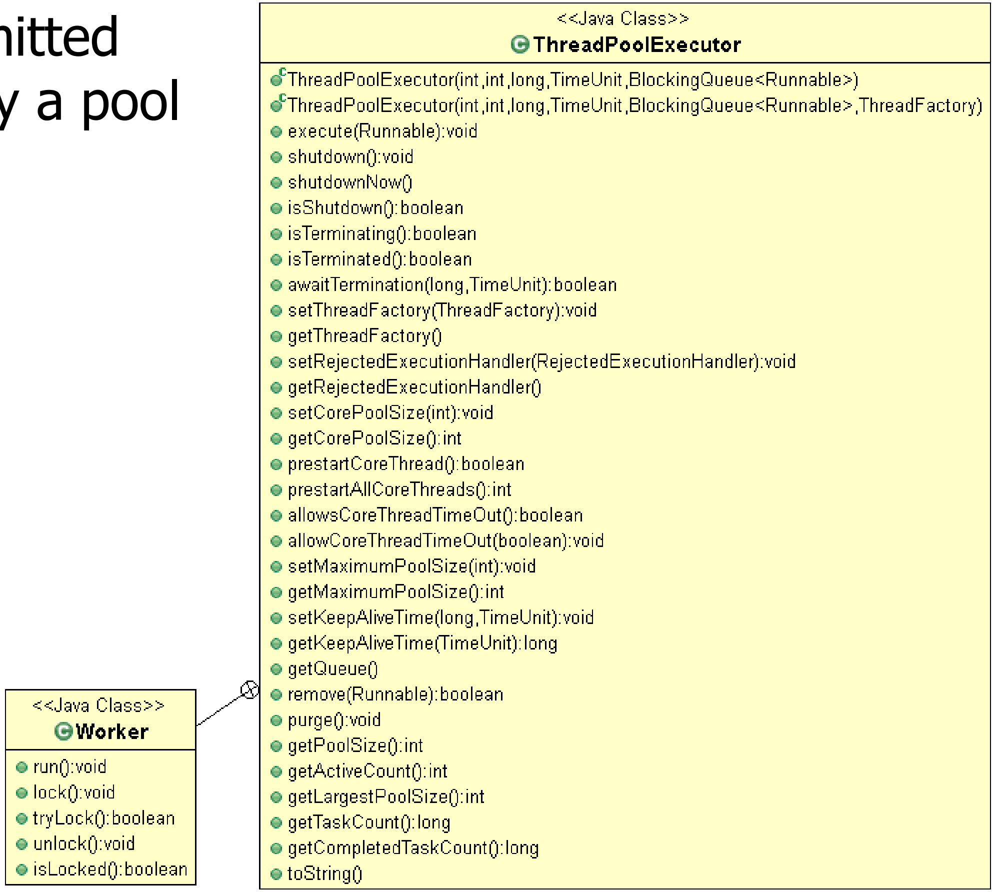
See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ThreadPoolExecutor.html

Overview of the Java ThreadPoolExecutor

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



A blocking queue passes tasks to threads



<<Java Class>>
ThreadPoolExecutor

- ThreadPoolExecutor(int,int,long,TimeUnit,BlockingQueue<Runnable>)
- ThreadPoolExecutor(int,int,long,TimeUnit,BlockingQueue<Runnable>,ThreadFactory)
- execute(Runnable):void
- shutdown():void
- shutdownNow()
- isShutdown():boolean
- isTerminating():boolean
- isTerminated():boolean
- awaitTermination(long,TimeUnit):boolean
- setThreadFactory(ThreadFactory):void
- getThreadFactory()
- setRejectedExecutionHandler(RejectedExecutionHandler):void
- getRejectedExecutionHandler()
- setCorePoolSize(int):void
- getCorePoolSize():int
- prestartCoreThread():boolean
- prestartAllCoreThreads():int
- allowsCoreThreadTimeOut():boolean
- allowCoreThreadTimeOut(boolean):void
- setMaximumPoolSize(int):void
- getMaximumPoolSize():int
- setKeepAliveTime(long,TimeUnit):void
- getKeepAliveTime(TimeUnit):long
- getQueue()
- remove(Runnable):boolean
- purge():void
- getPoolSize():int
- getActiveCount():int
- getLargestPoolSize():int
- getTaskCount():long
- getCompletedTaskCount():long
- toString()

<<Java Class>>
Worker

- run():void
- lock():void
- tryLock():boolean
- unlock():void
- isLocked():boolean

Overview of the Java ThreadPoolExecutor

- ThreadPoolExecutor's constructor can be configured via various parameters

```
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

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

Overview of the Java ThreadPoolExecutor

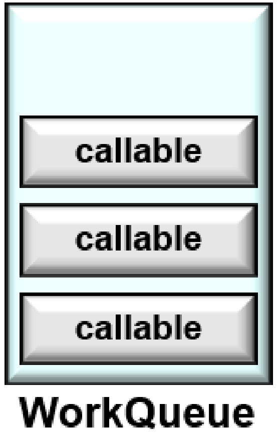
- The lifetime of threads in the pool can be controlled programmatically
  - **`keepAliveTime`** – maximum time that excess idle threads will wait for new tasks before terminating when # of threads is greater than the core
  - **`unit`** – the time unit for the **`keepAliveTime`** argument

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

Overview of the Java ThreadPoolExecutor

- 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

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



Overview of the Java ThreadPoolExecutor

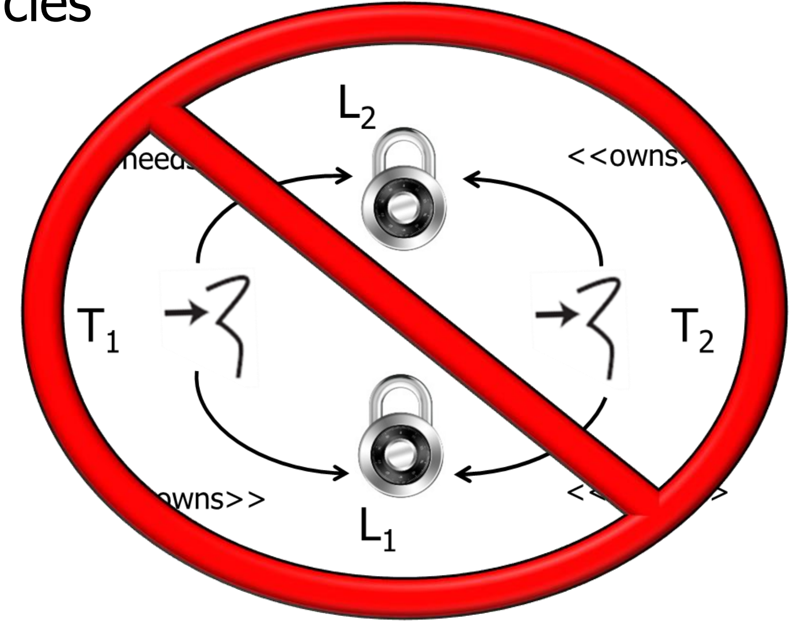
- The queue can be strategized
  - Direct handoff (used by cached pool)

```
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/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

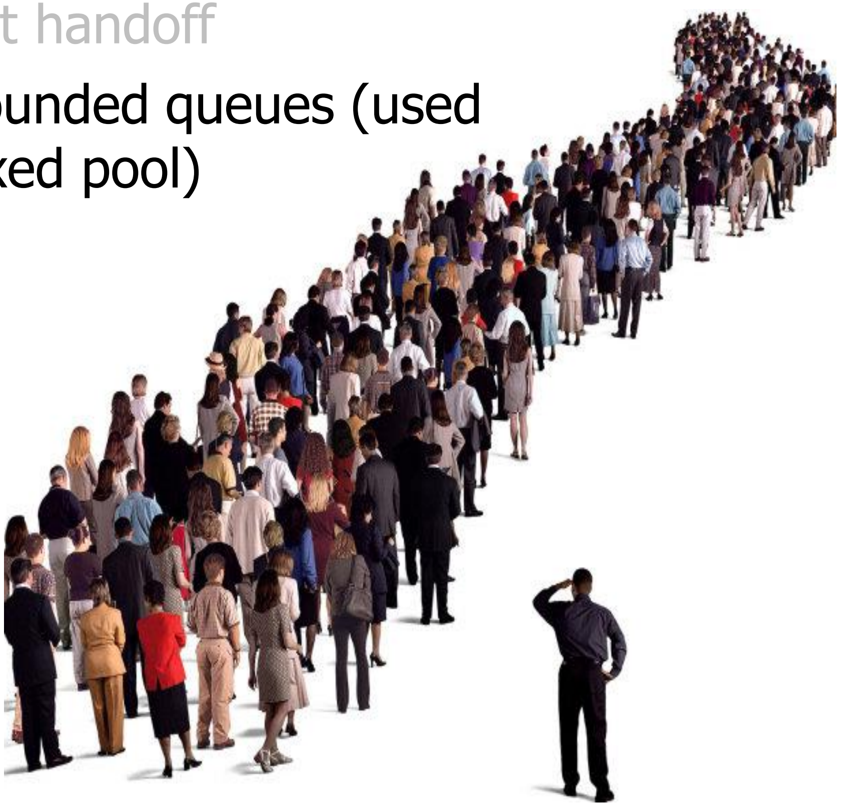
Overview of the Java ThreadPoolExecutor

- 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)



```
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

```
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

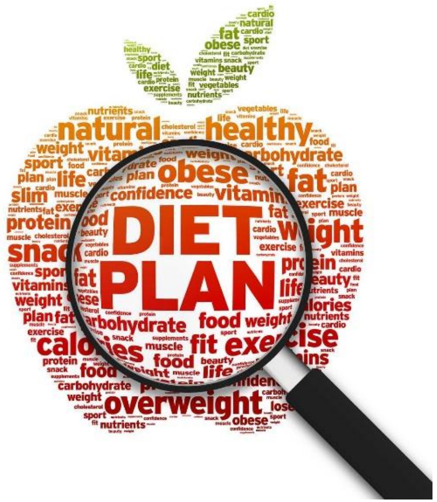
- The queue can be strategized
  - Direct handoff
  - Unbounded queues
  - Bounded queues (also used by fixed pool)

```
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/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

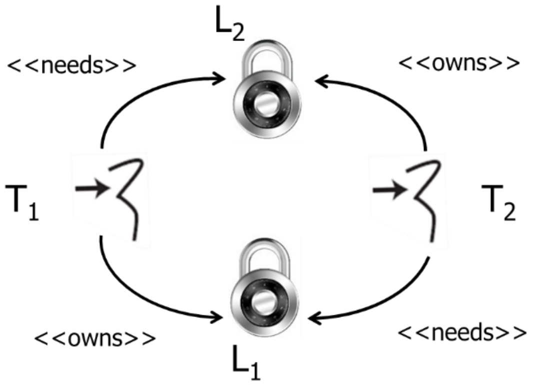


```
ThreadPoolExecutor
          (int corePoolSize,
           int maximumPoolSize,
           long keepAliveTime,
           TimeUnit unit,
           BlockingQueue<Runnable>
               workQueue,
           ThreadFactory
               threadFactory)
```

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

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

```
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