The Java ExecutorService Interface (Part 3)

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

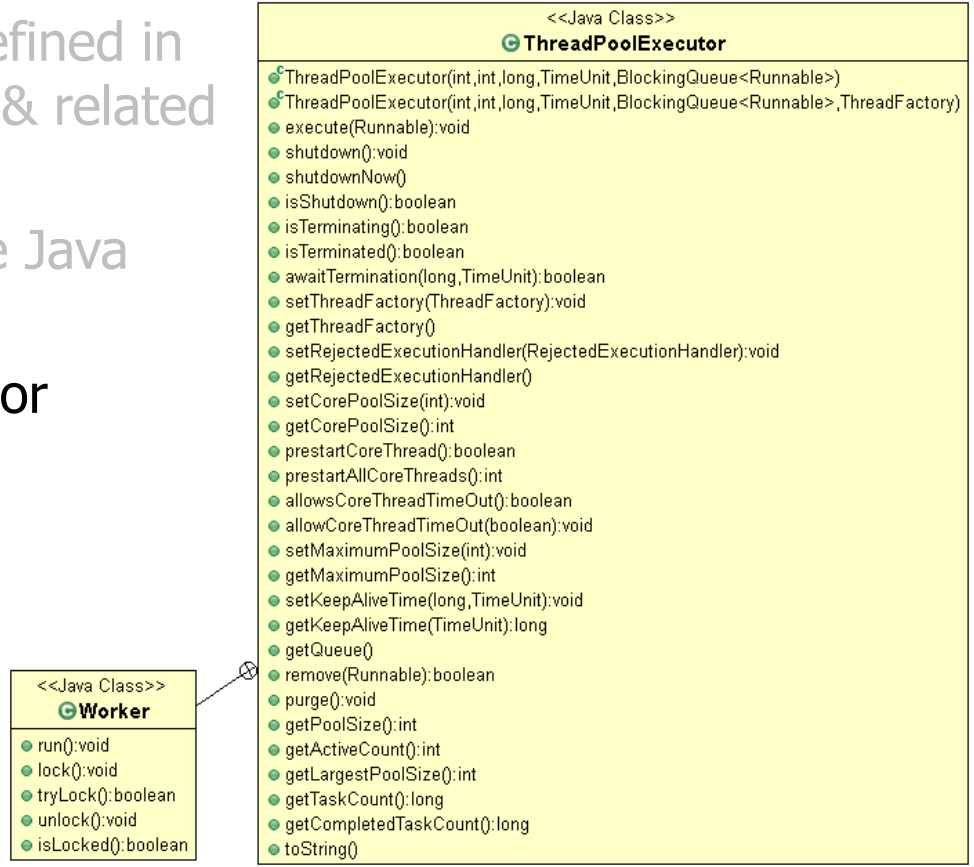
Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

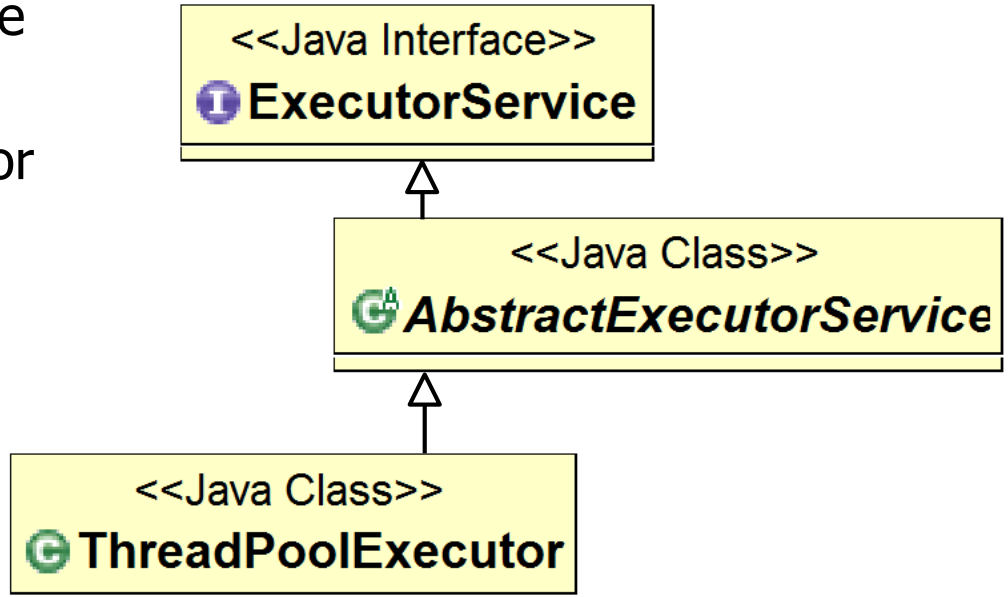
- Recognize the powerful features defined in the Java ExecutorService interface & related interfaces/classes
- Know key methods provided by the Java ExecutorService
- Understand how ThreadPoolExecutor implements the ExecutorService



Overview of the Java ThreadPoolExecutor

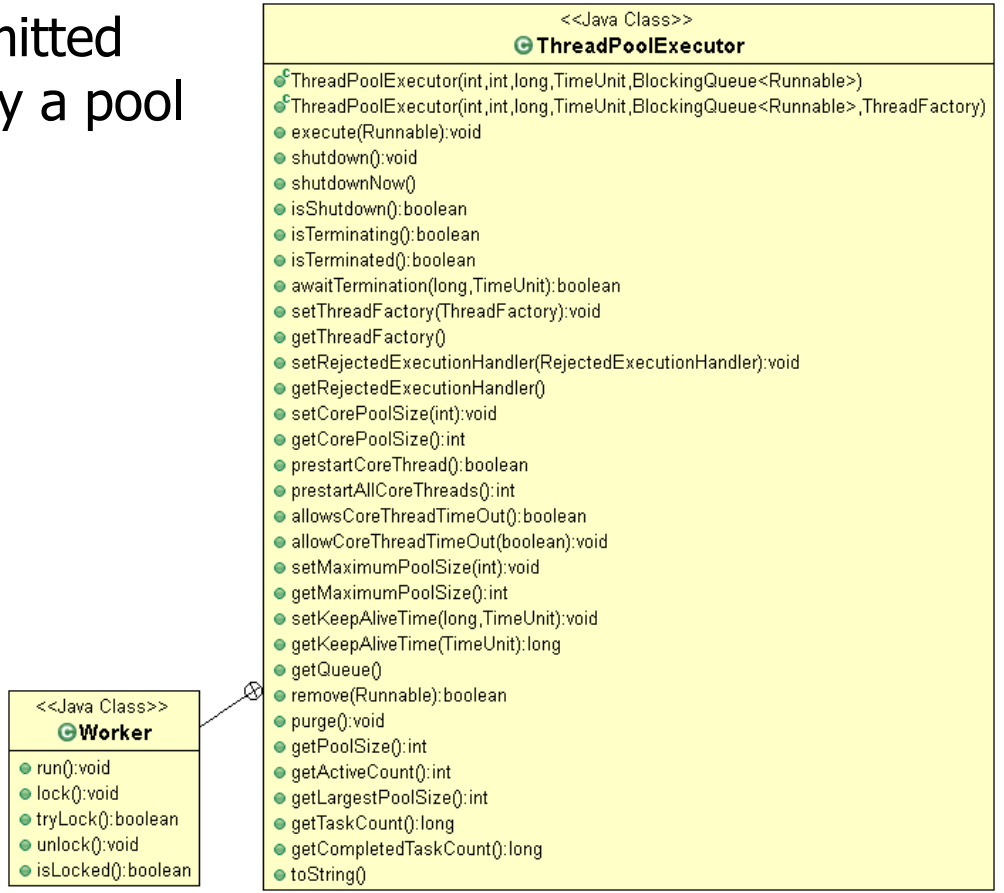
Overview of the Java ThreadPoolExecutor

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



Overview of the Java ThreadPoolExecutor

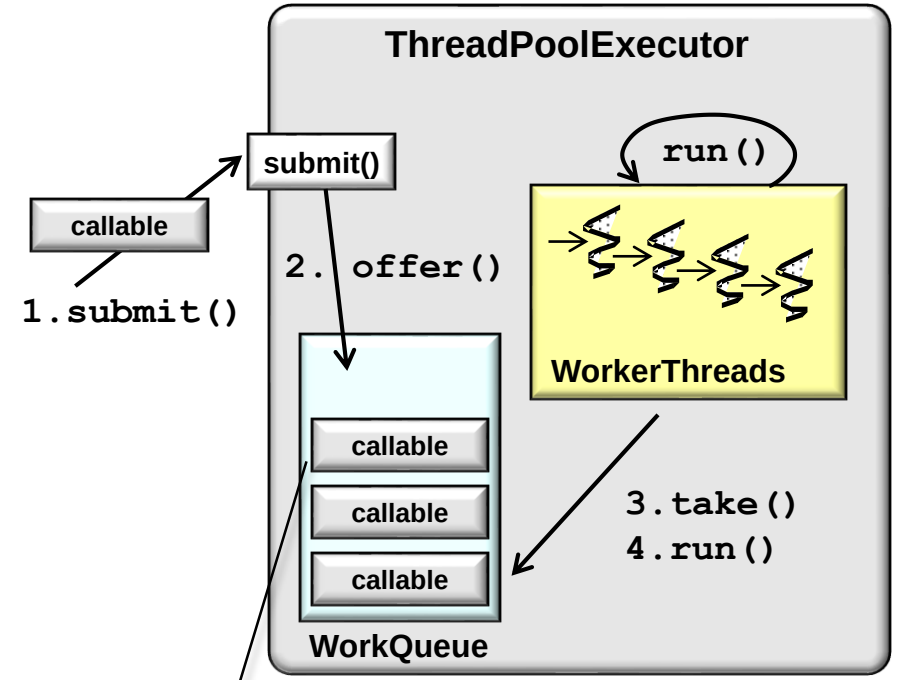
- ThreadPoolExecutor runs each submitted task via a worker thread provided by a pool



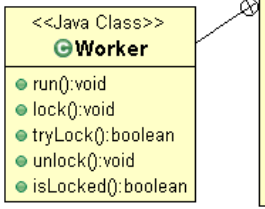
See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ThreadPoolExecutor.html

Overview of the Java ThreadPoolExecutor

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

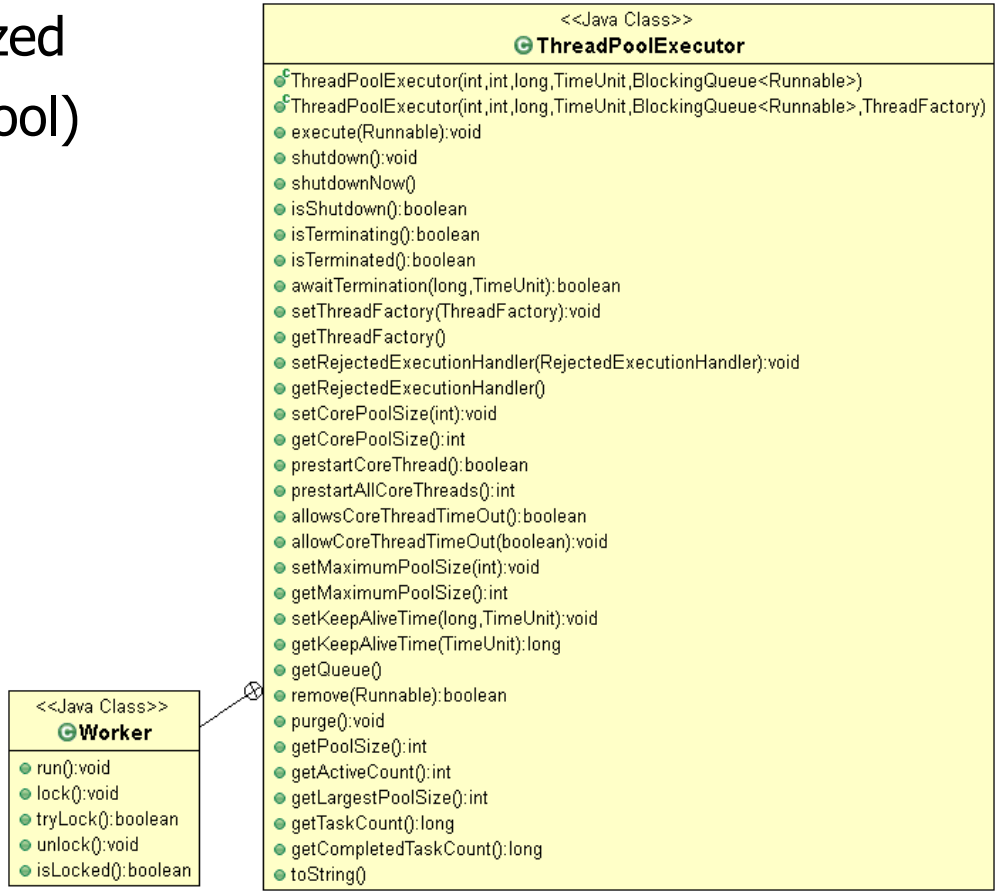


A blocking queue passes tasks to threads



Overview of the Java ThreadPoolExecutor

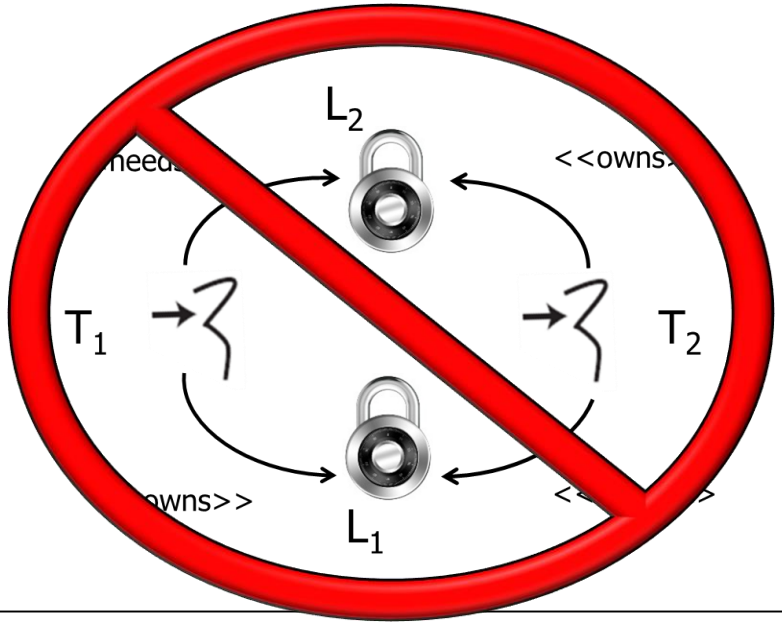
- The blocking queue can be strategized
  - Direct handoff (used by cached pool)

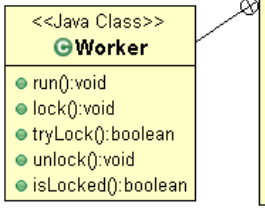


See docs.oracle.com/javase/8/docs/api/java/util/concurrent/SynchronousQueue.html

Overview of the Java ThreadPoolExecutor

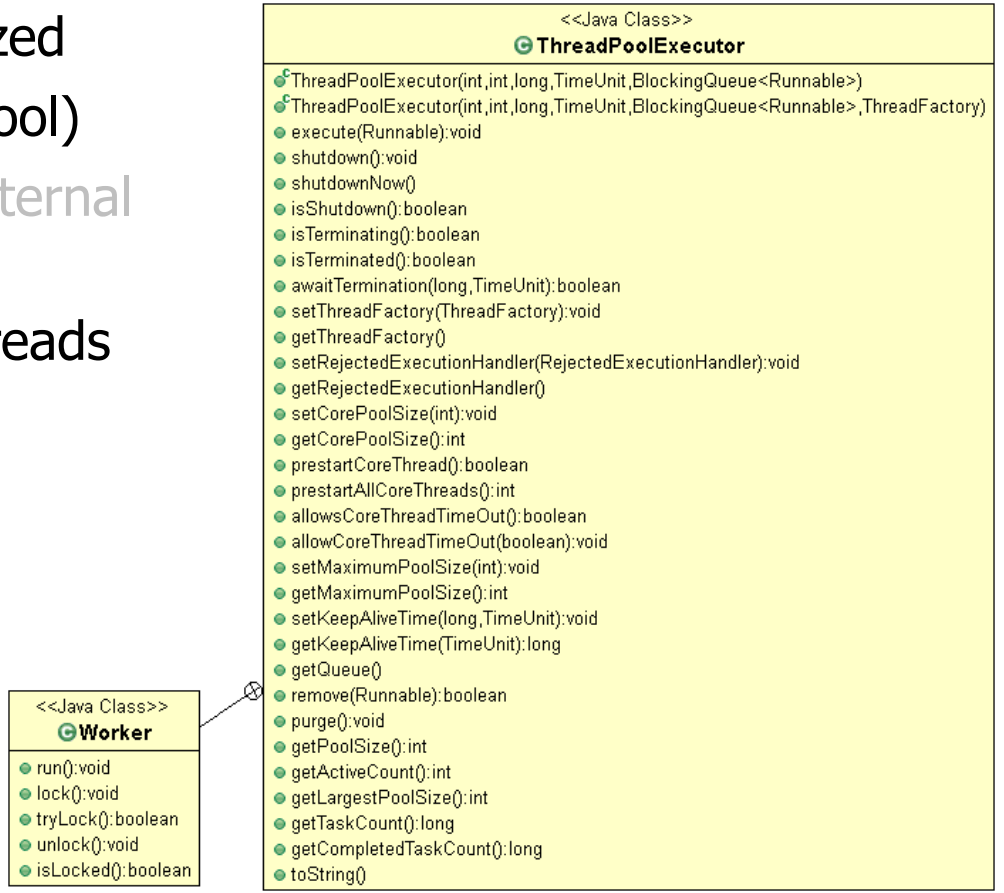
- The blocking queue can be strategized
  - Direct handoff (used by cached pool)
    - Pros – Avoids deadlock when internal dependencies





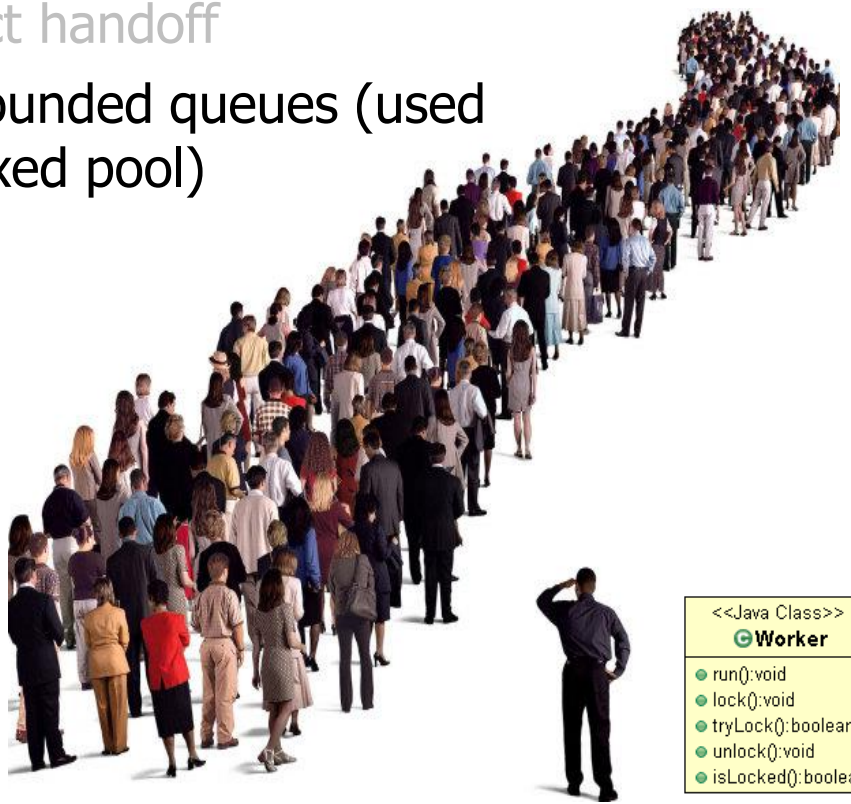
Overview of the Java ThreadPoolExecutor

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



Overview of the Java ThreadPoolExecutor

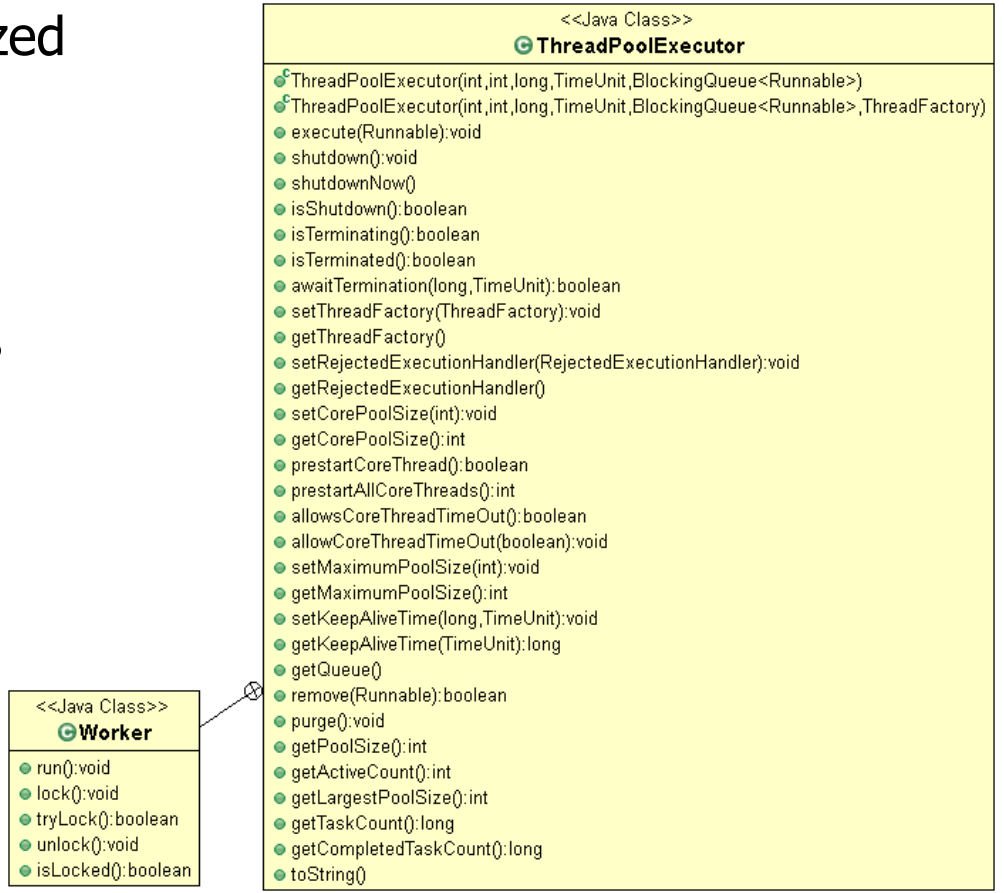
- The blocking queue can be strategized
  - Direct handoff
  - Unbounded queues (used by fixed pool)



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

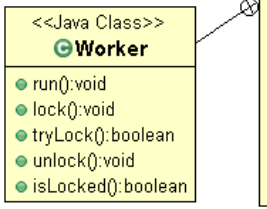
Overview of the Java ThreadPoolExecutor

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



Overview of the Java ThreadPoolExecutor

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



Overview of the Java ThreadPoolExecutor

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

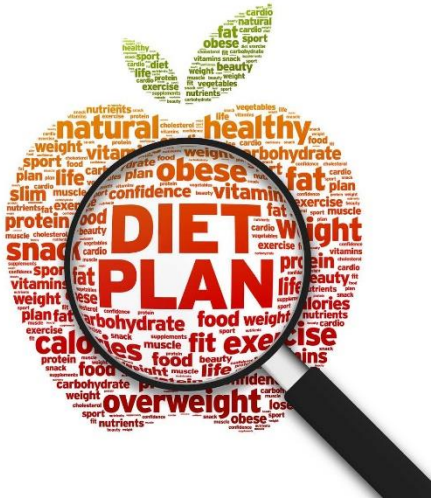
<<Java Class>>
Worker

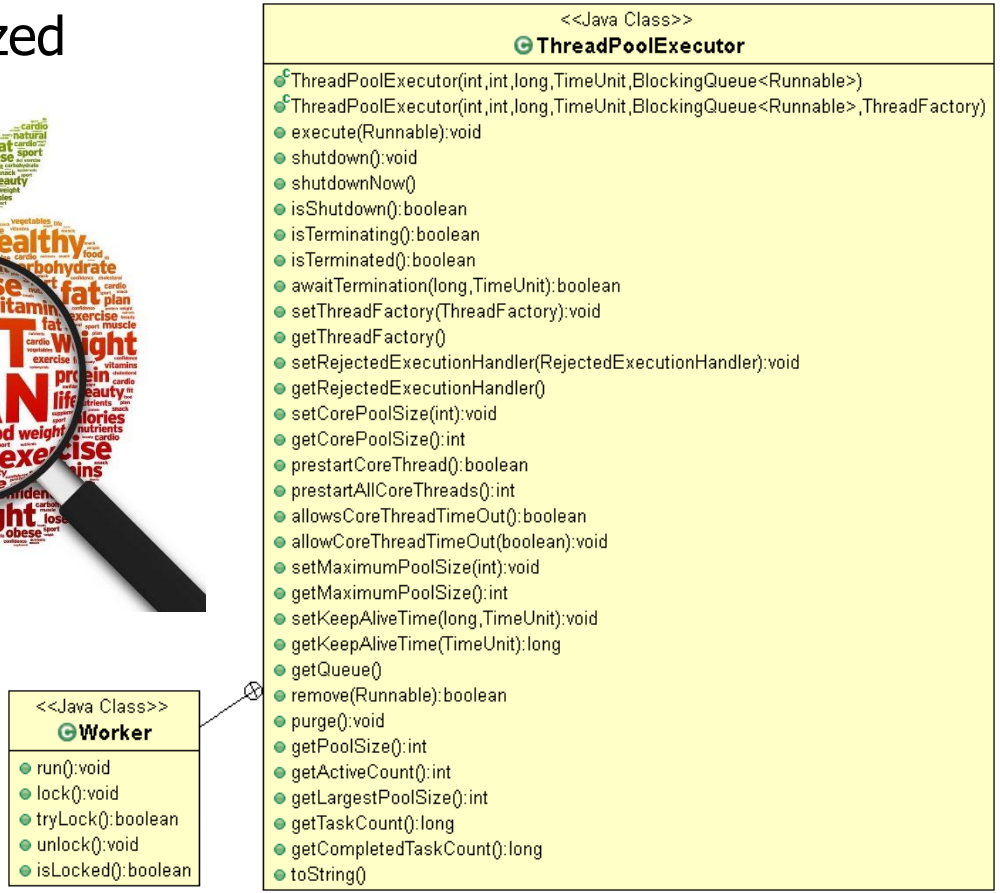
- run():void
- lock():void
- tryLock():boolean
- unlock():void
- isLocked():boolean

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

Overview of the Java ThreadPoolExecutor

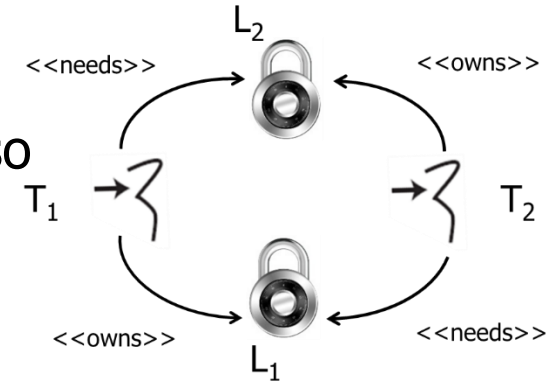
- The blocking queue can be strategized
  - Direct handoff
  - Unbounded queues
  - Bounded queues (also used by fixed pool)
    - Pros – Limits resource utilization

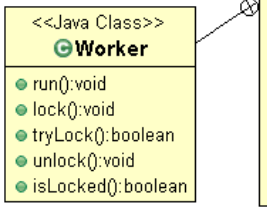




Overview of the Java ThreadPoolExecutor

- The blocking 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

End of The JavaExecutor Service (Part 3)