The Java Executor Framework: Introduction

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

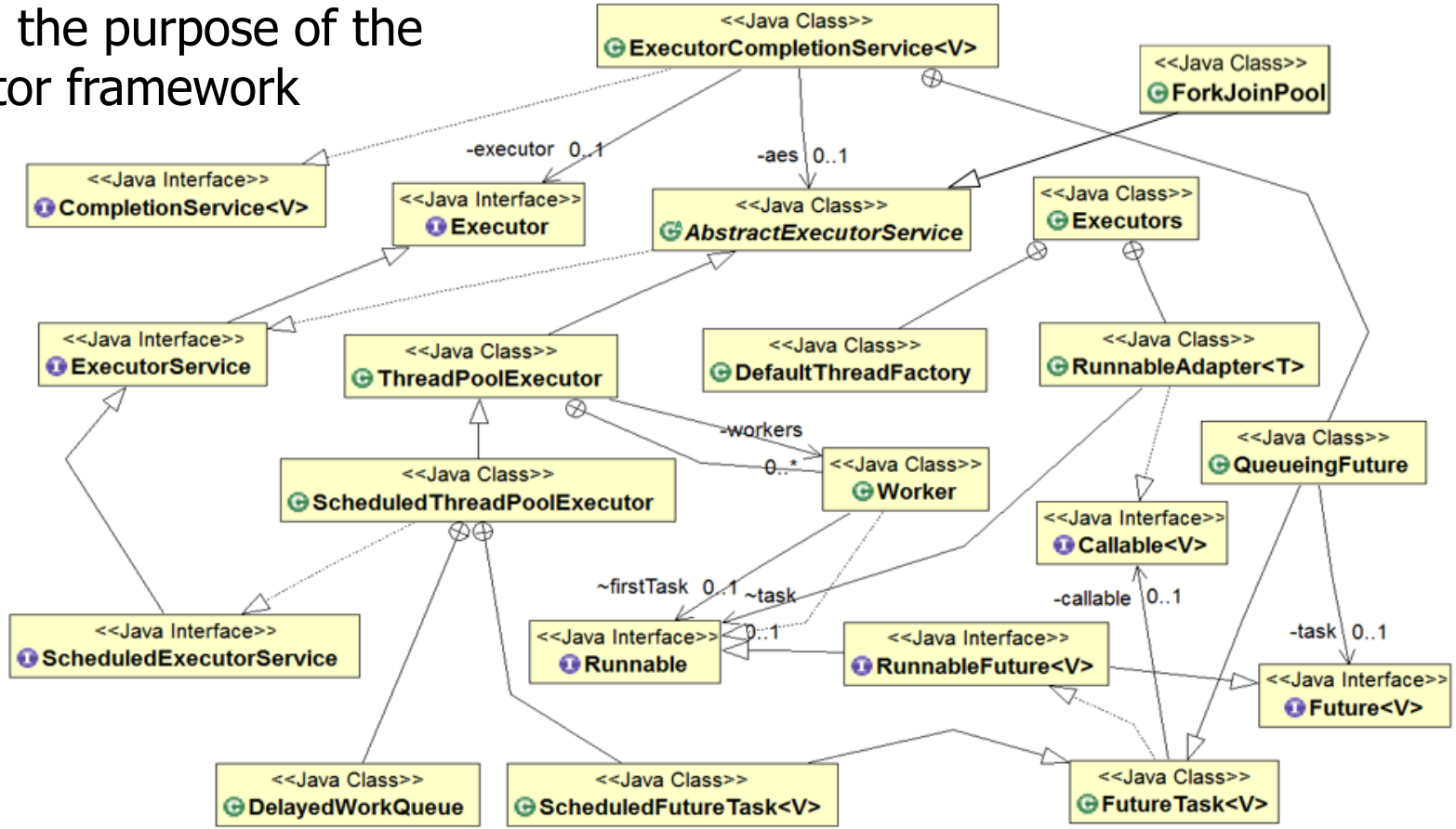
Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand the purpose of the Java executor framework

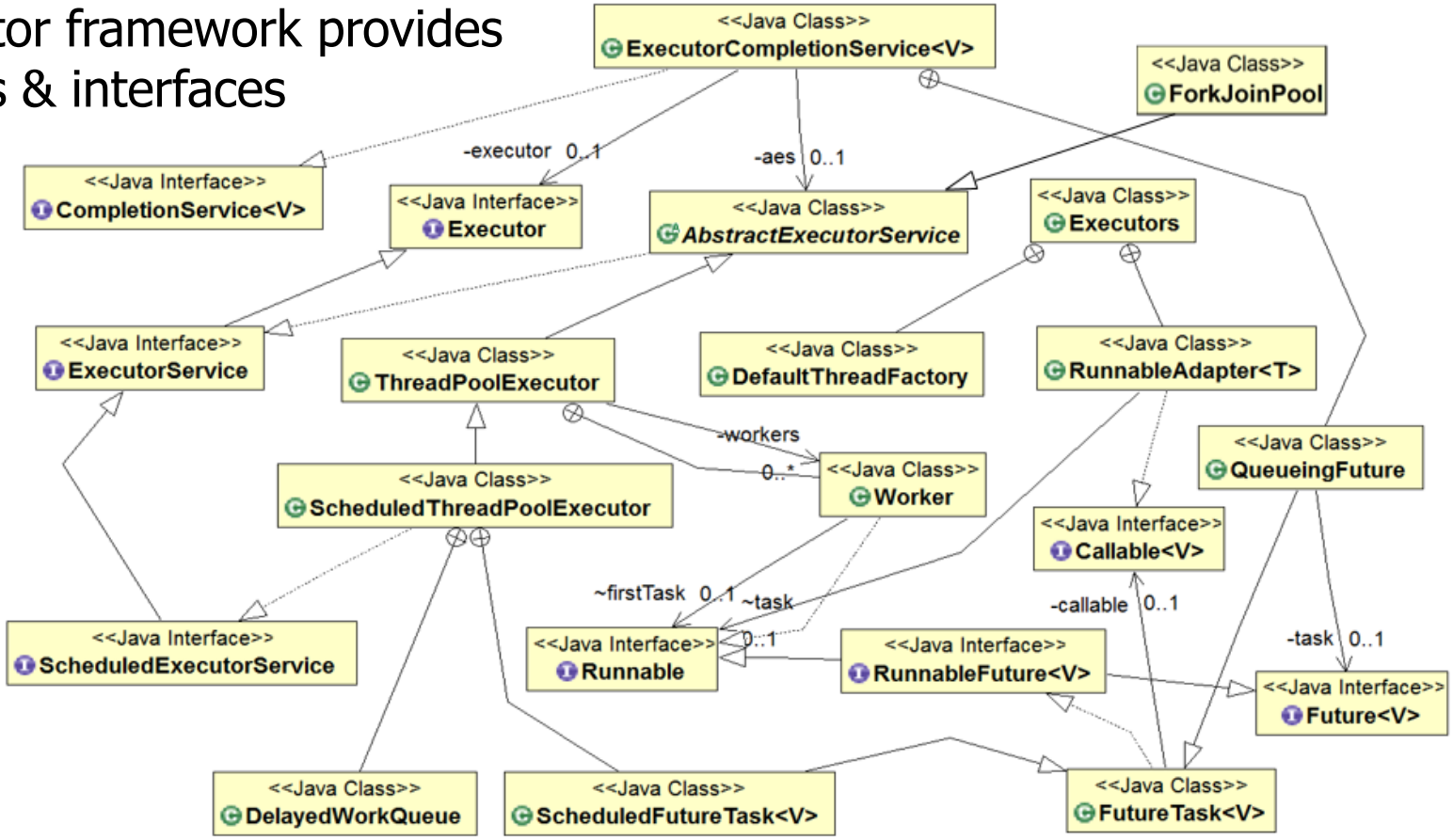


See docs.oracle.com/javase/tutorial/essential/concurrency/executors.html

Overview of the Java Executor Framework

Overview of The Java Executor Framework

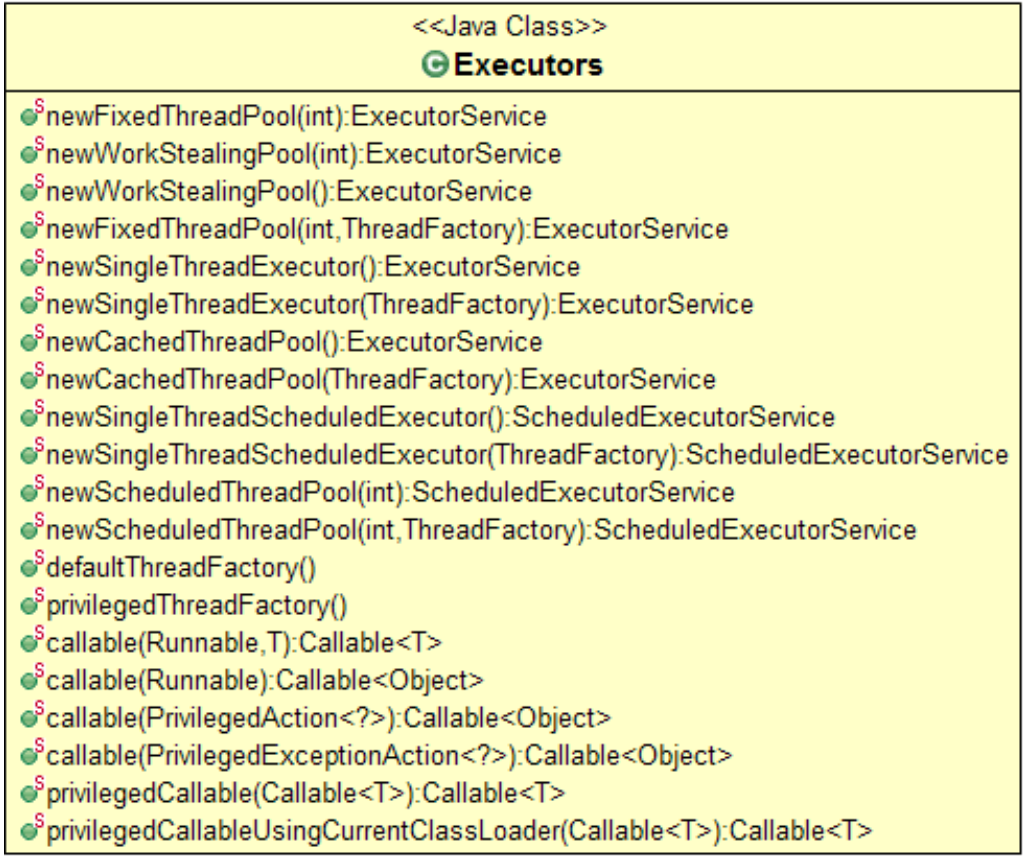
- Java's executor framework provides many classes & interfaces



Decouples thread creation & management from the rest of the app logic

Overview of The Java Executor Framework

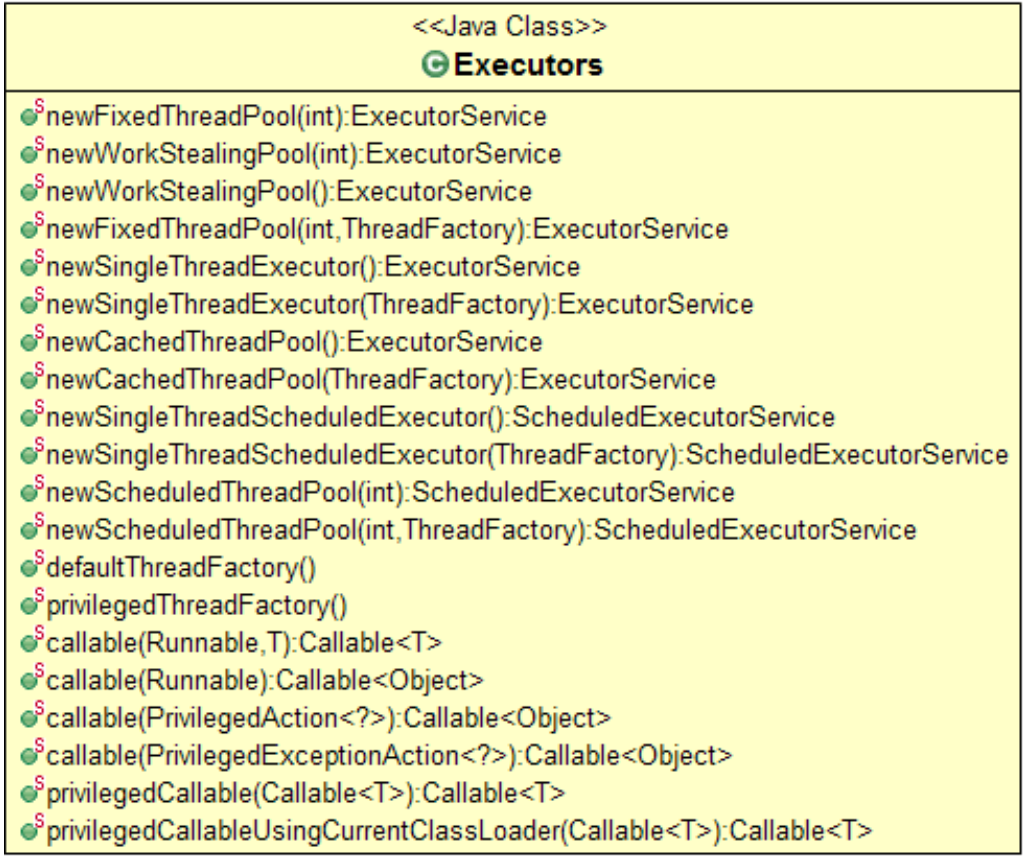
- The Executors utility class provides access to key capabilities in the Java executor framework



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

Overview of The Java Executor Framework

- The Executors utility class provides access to key capabilities in the Java executor framework
  - A utility class is final w/only static methods, no (non-static) state, & a private constructor



See www.quora.com/What-is-the-best-way-to-write-utility-classes-in-Java/answer/Jon-Harley

Overview of The Java Executor Framework

- The Executors utility class provides access to key capabilities in the Java executor framework
  - A utility class is final w/only static methods, no (non-static) state, & a private constructor
  - Its factory methods create various types of thread pools

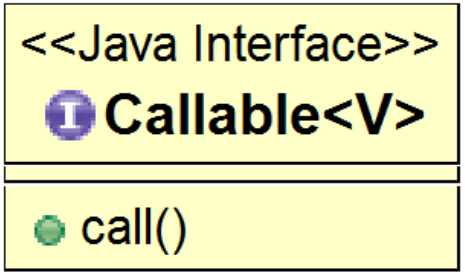
<<Java Class>>
Executors

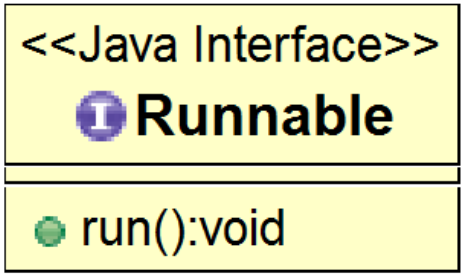
- newFixedThreadPool(int):ExecutorService
- newWorkStealingPool(int):ExecutorService
- newWorkStealingPool():ExecutorService
- newFixedThreadPool(int,ThreadFactory):ExecutorService
- newSingleThreadExecutor():ExecutorService
- newSingleThreadExecutor(ThreadFactory):ExecutorService
- newCachedThreadPool():ExecutorService
- newCachedThreadPool(ThreadFactory):ExecutorService
- newSingleThreadScheduledExecutor():ScheduledExecutorService
- newSingleThreadScheduledExecutor(ThreadFactory):ScheduledExecutorService
- newScheduledThreadPool(int):ScheduledExecutorService
- newScheduledThreadPool(int,ThreadFactory):ScheduledExecutorService
- defaultThreadFactory()
- privilegedThreadFactory()
- callable(Runnable,T):Callable<T>
- callable(Runnable):Callable<Object>
- callable(PrivilegedAction<?>):Callable<Object>
- callable(PrivilegedExceptionAction<?>):Callable<Object>
- privilegedCallable(Callable<T>):Callable<T>
- privilegedCallableUsingCurrentClassLoader(Callable<T>):Callable<T>

See en.wikipedia.org/wiki/Thread_pool_pattern

Overview of The Java Executor Framework

- The Executors utility class provides access to key capabilities in the Java executor framework
  - A utility class is final w/only static methods, no (non-static) state, & a private constructor
  - Its factory methods create various types of thread pools
- A thread pool can execute one-way or two-way tasks concurrently on multiple processor cores





See www.javaworld.com/article/2071822/book-excerpt--executing-tasks-in-threads.html

Overview of The Java Executor Framework

- The Executors utility class provides access to key capabilities in the Java executor framework
  - A utility class is final w/only static methods, no (non-static) state, & a private constructor
  - Its factory methods create various types of thread pools
  - A thread pool can execute one-way or two-way tasks concurrently on multiple processor cores
    - A task is a logical unit of work that (ideally) doesn't depend on the state, result, or side effects of other tasks

See www.javaworld.com/article/2071822/book-excerpt--executing-tasks-in-threads.html

End of the Java Executor Framework: Introduction