

Key Methods in the Java ForkJoinTask Superclass

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

**Institute for Software
Integrated Systems**

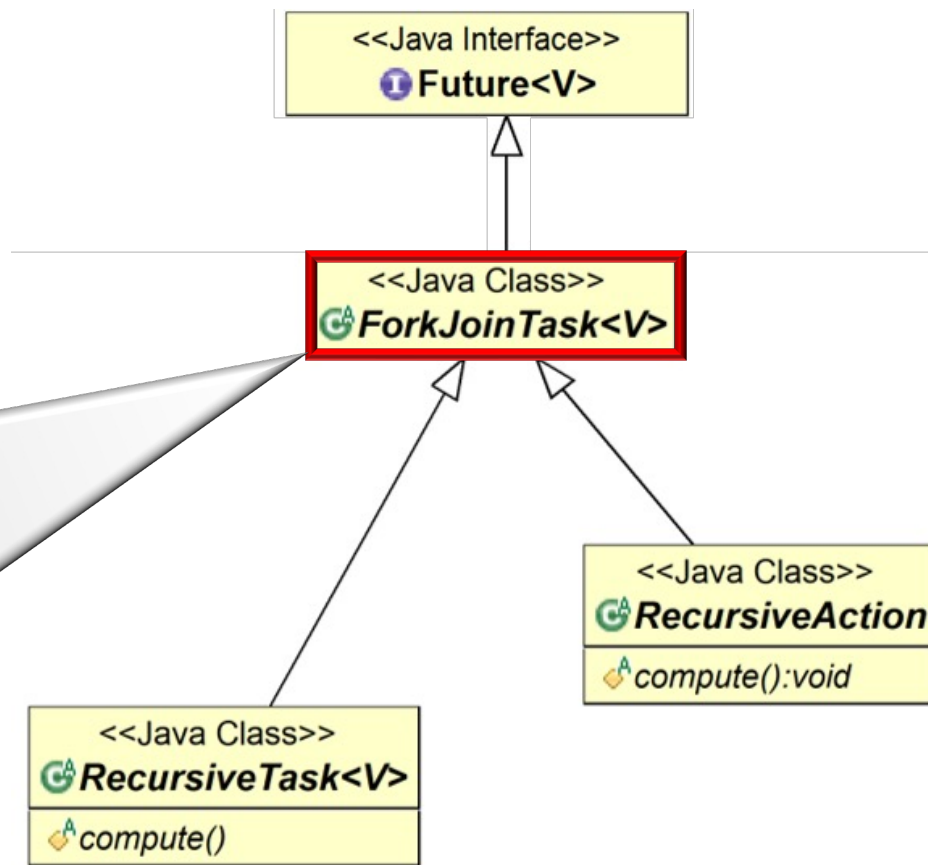
**Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Part of the Lesson

- Recognize the key methods in the ForkJoinPool class
- Recognize the key methods in the ForkJoinTask class

```
ForkJoinTask<V>  
m ForkJoinTask()  
m cancel(boolean) boolean  
m complete(V) void  
m fork() ForkJoinTask<V>  
m get(long, TimeUnit) V  
m get() V  
m invoke() V  
m invokeAll(ForkJoinTask<?>[]) void  
m invokeAll(Collection<T>) Collection<T>  
m invokeAll(ForkJoinTask<?>, ForkJoinTask<?>) void  
m isCancelled() boolean  
m isDone() boolean  
m join() V  
m resultNow() V
```

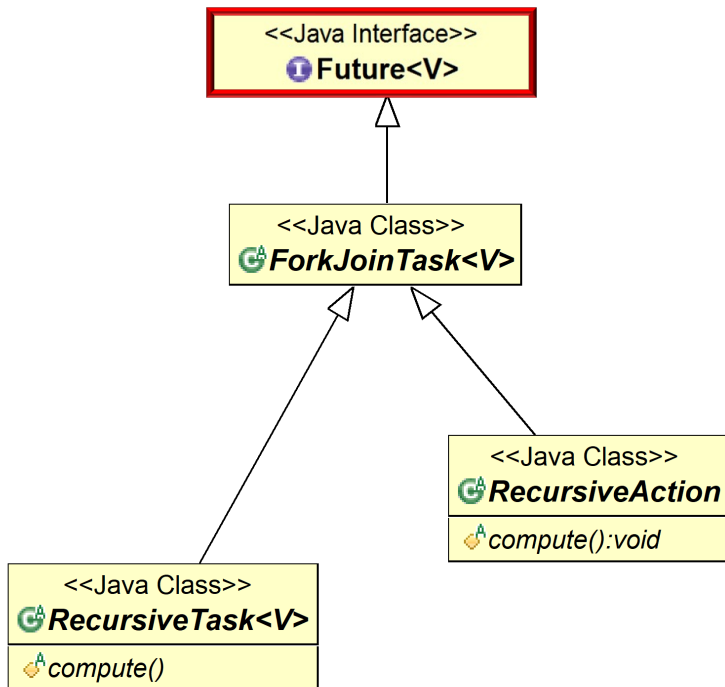


Key Methods in Java ForkJoinTask

Key Methods in Java ForkJoinTask

- ForkJoinTask implements Future

```
abstract class ForkJoinTask<V>  
    implements Future<V>,  
        Serializable {  
  
    ...  
}
```

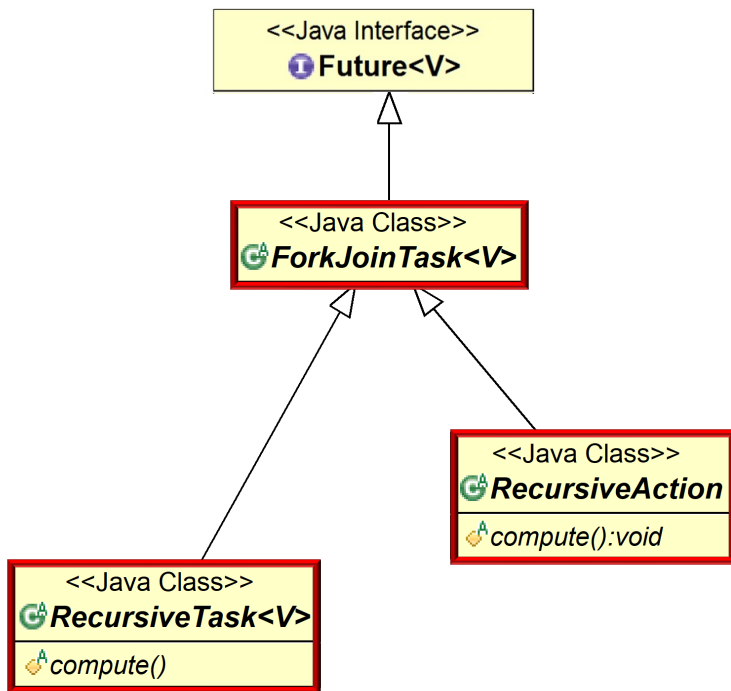


See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinTask.html

Key Methods in Java ForkJoinTask

- ForkJoinTask implements Future

```
abstract class ForkJoinTask<V>  
    implements Future<V>,  
        Serializable {  
  
    ...  
}
```



It's uncommon to use these future methods, but rather use subclass methods

Key Methods in Java ForkJoinTask

- There are three key methods
 - `fork()` enables a task to create sub-tasks that run in parallel

```
abstract class ForkJoinTask<V>
    implements Future<V>,
               Serializable {
    ...
    final ForkJoinTask<V> fork()
    { ... }

    final V join() { ... }

    final V invoke() { ... }
```

Key Methods in Java ForkJoinTask

- There are three key methods
 - fork() enables a task to create sub-tasks that run in parallel
 - Arrange to execute this task asynchronously in the current task's pool or ForkJoinPool's common pool



```
abstract class ForkJoinTask<V>
    implements Future<V>,
        Serializable {
    ...
    final ForkJoinTask<V> fork()
    { ... }

    final V join() { ... }

    final V invoke() { ... }
```

Key Methods in Java ForkJoinTask

- There are three key methods
 - fork() enables a task to create sub-tasks that run in parallel
 - Arrange to execute this task asynchronously in the current task's pool or ForkJoinPool's common pool



```
abstract class ForkJoinTask<V>
    implements Future<V>,
        Serializable {
    ...
    final ForkJoinTask<V> fork()
    { ... }

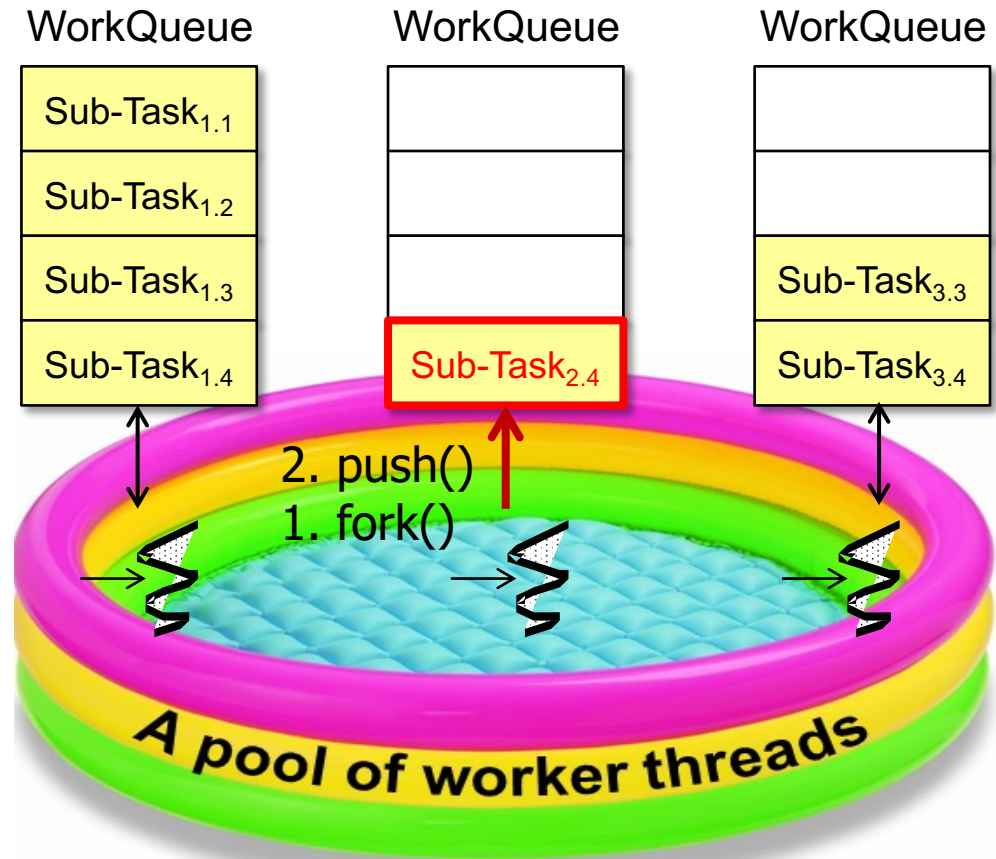
    final V join() { ... }

    final V invoke() { ... }
```

The fork() method does not block the caller

Key Methods in Java ForkJoinTask

- There are three key methods
 - `fork()` enables a task to create sub-tasks that run in parallel
 - Arrange to execute this task asynchronously in the current task's pool or `ForkJoinPool`'s common pool
 - Pushes the task on the *head* of the deque owned by the current worker thread



Key Methods in Java ForkJoinTask

- There are three key methods
 - `fork()` enables a task to create sub-tasks that run in parallel
 - `join()` returns the result of a previously fork'd computation when it's done

```
abstract class ForkJoinTask<V>
    implements Future<V>,
               Serializable {
    ...
    final ForkJoinTask<V> fork()
    { ... }

    final V join() { ... }

    final V invoke() { ... }
```

Key Methods in Java ForkJoinTask

- There are three key methods
 - `fork()` enables a task to create sub-tasks that run in parallel
 - `join()` returns the result of a previously fork'd computation when it's done
 - Calling task is "blocked" until forked sub-task is done

```
abstract class ForkJoinTask<V>
    implements Future<V>,
        Serializable {
    ...
    final ForkJoinTask<V> fork()
    { ... }

    final V join() { ... }

    final V invoke() { ... }
```



Key Methods in Java ForkJoinTask

- There are three key methods
 - fork() enables a task to create sub-tasks that run in parallel
 - join() returns the result of a previously fork'd computation when it's done
 - Calling task is "blocked" until forked sub-task is done

```
abstract class ForkJoinTask<V>
    implements Future<V>,
        Serializable {
    ...
    final ForkJoinTask<V> fork()
    { ... }

    final V join() { ... }

    final V invoke() { ... }
```



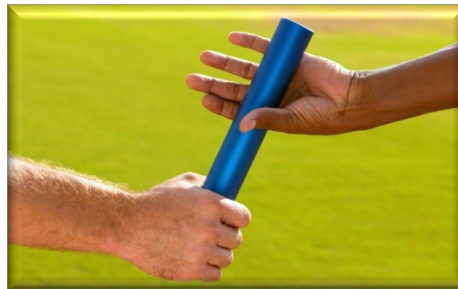
*"Collaborative Jiffy Lube"
model of processing!*

See en.wikipedia.org/wiki/Jiffy_Lube

Key Methods in Java ForkJoinTask

- There are three key methods
 - `fork()` enables a task to create sub-tasks that run in parallel
 - `join()` returns the result of a previously fork'd computation when it's done
 - Calling task is "blocked" until forked sub-task is done
 - Defines a synchronization point

```
abstract class ForkJoinTask<V>  
    implements Future<V>,  
                Serializable {  
    ...  
    final ForkJoinTask<V> fork()  
    { ... }  
  
    final V join() { ... }  
  
    final V invoke() { ... }
```



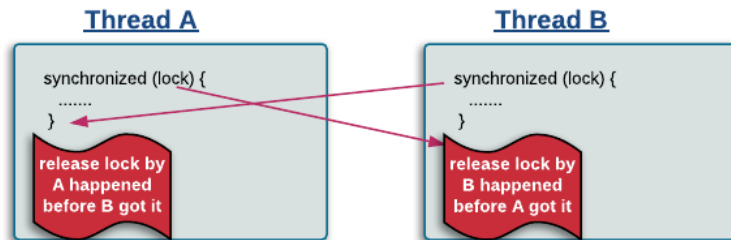
Key Methods in Java ForkJoinTask

- There are three key methods
 - `fork()` enables a task to create sub-tasks that run in parallel
 - `join()` returns the result of a previously fork'd computation when it's done
 - Calling task is "blocked" until forked sub-task is done
 - Defines a synchronization point
 - Ensures all writes in a worker thread that "happen-before" `join()` are made visible to other threads after the `join()`

```
abstract class ForkJoinTask<V>
    implements Future<V>,
               Serializable {
    ...
    final ForkJoinTask<V> fork()
    { ... }

    final V join() { ... }

    final V invoke() { ... }
```



Key Methods in Java ForkJoinTask

- There are three key methods
 - `fork()` enables a task to create sub-tasks that run in parallel
 - `join()` returns the result of a previously fork'd computation when it's done
 - `invoke()` performs this task, awaits its completion if needed, & returns its result

```
abstract class ForkJoinTask<V>
    implements Future<V>,
               Serializable {
    ...
    final ForkJoinTask<V> fork()
    { ... }

    final V join() { ... }

    final V invoke() { ... }
```



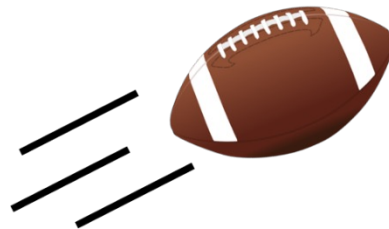
Key Methods in Java ForkJoinTask

- There are three key methods
 - `fork()` enables a task to create sub-tasks that run in parallel
 - `join()` returns the result of a previously fork'd computation when it's done
- `invoke()` performs this task, awaits its completion if needed, & returns its result
 - Throws `RuntimeException` or `Error` if the underlying computation did so

```
abstract class ForkJoinTask<V>
    implements Future<V>,
               Serializable {
    ...
    final ForkJoinTask<V> fork()
    { ... }

    final V join() { ... }

    final V invoke() { ... }
```



End of Key Methods in the Java ForkJoinTask Superclass