

Key Methods in the Java Recursive Action & RecursiveTask Subclasses

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt



Professor of Computer Science

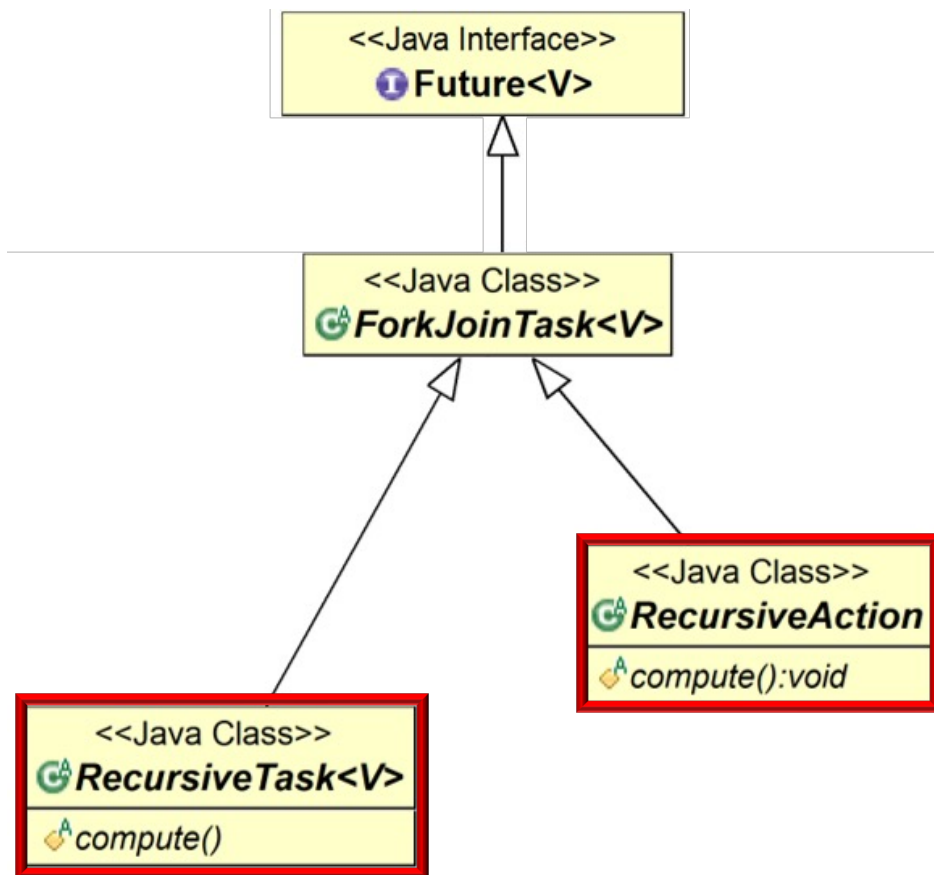
**Institute for Software
Integrated Systems**

**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
- Recognize the key methods in the RecursiveAction & RecursiveTask classes

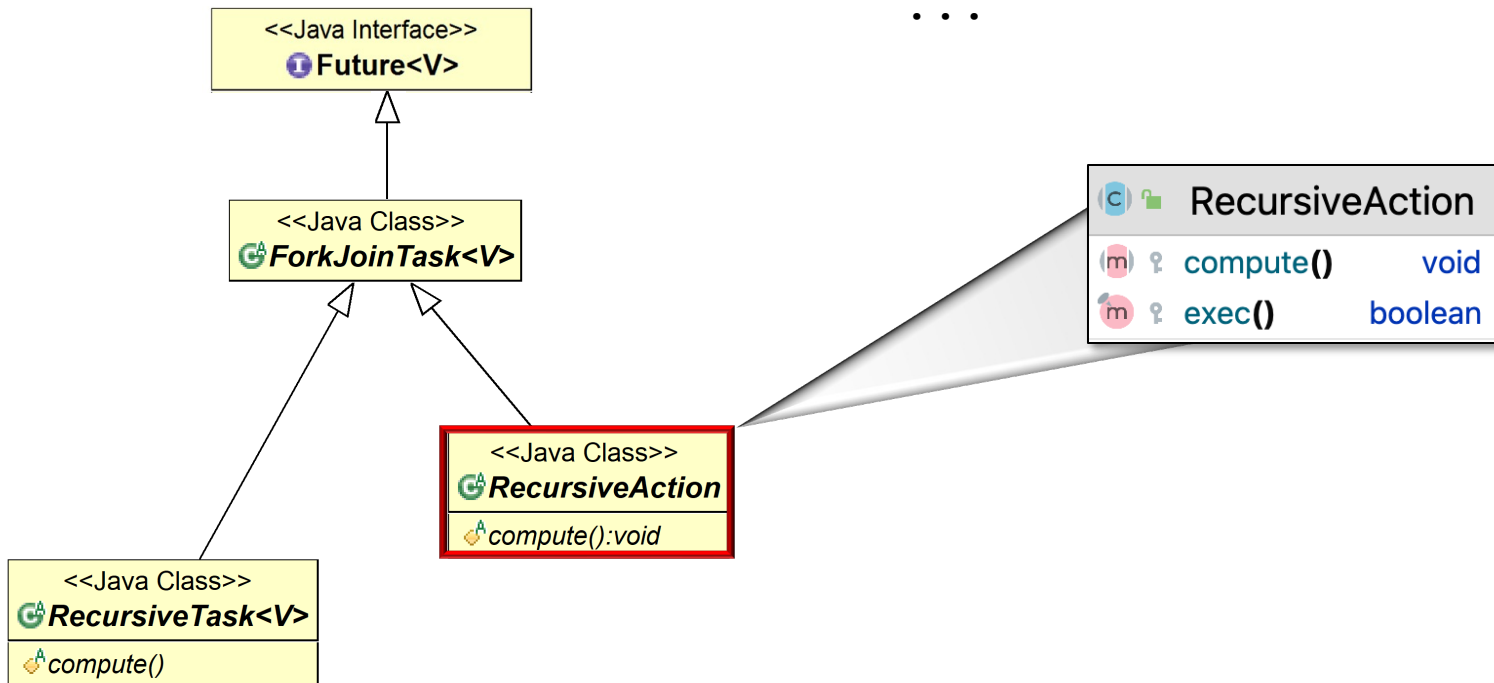


Key Methods in the Java RecursiveAction

Key Methods in Java RecursiveAction

- RecursiveAction extends ForkJoinTask & does not return a result

```
abstract class RecursiveAction  
    extends ForkJoinTask<Void> {  
    ...
```

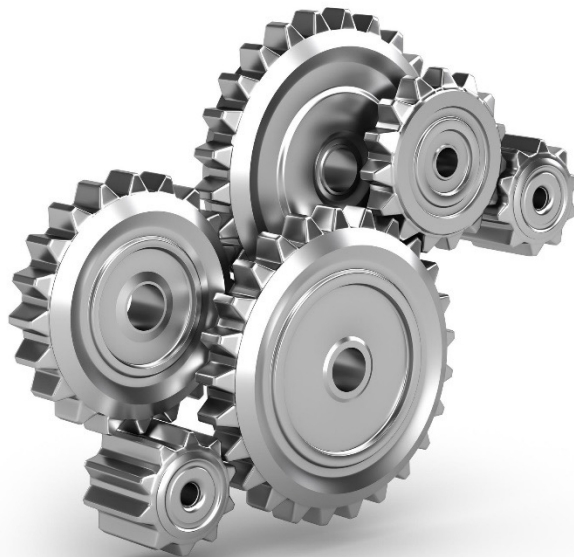


See docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveAction.html

Key Methods in Java RecursiveAction

- RecursiveAction extends ForkJoinTask & does not return a result
- Subclasses override compute() to perform task's main computation

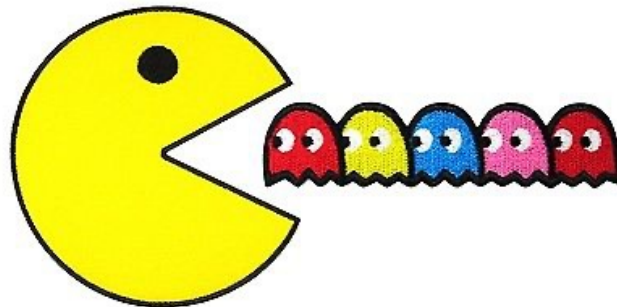
```
abstract class RecursiveAction  
    extends ForkJoinTask<Void> {  
    protected abstract Void  
        compute();  
    ...  
}
```



Key Methods in Java RecursiveAction

- RecursiveAction extends ForkJoinTask & does not return a result
- Subclasses override compute() to perform task's main computation
 - If data size is below a certain threshold perform work directly

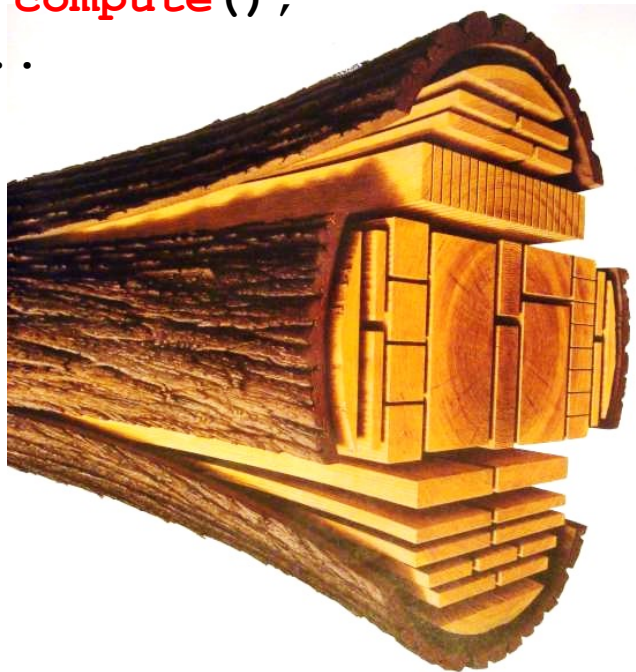
```
abstract class RecursiveAction
    extends ForkJoinTask<Void> {
    protected abstract Void
        compute();
    ...
}
```



Key Methods in Java RecursiveAction

- RecursiveAction extends ForkJoinTask & does not return a result
- Subclasses override compute() to perform task's main computation
 - If data size is below a certain threshold perform work directly
 - If data size is large, split work into smaller sub-tasks that are fork()'d to run in parallel

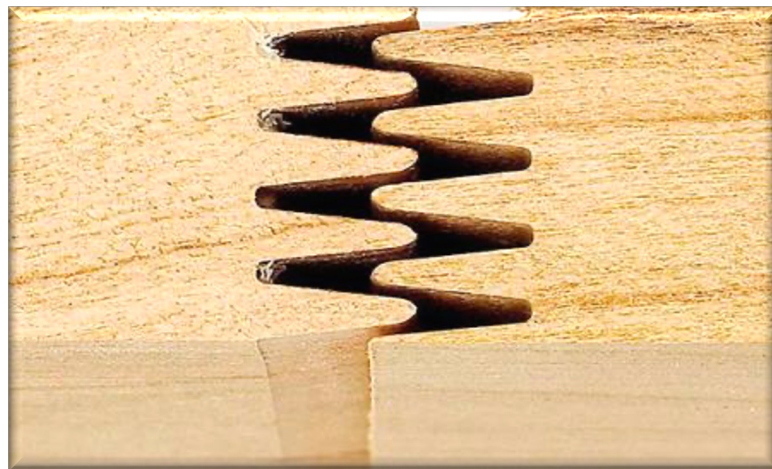
```
abstract class RecursiveAction  
    extends ForkJoinTask<Void> {  
    protected abstract Void  
        compute();  
    ...  
}
```



Key Methods in Java RecursiveAction

- RecursiveAction extends ForkJoinTask & does not return a result
- Subclasses override compute() to perform task's main computation
 - If data size is below a certain threshold perform work directly
 - If data size is large, split work into smaller sub-tasks that are fork()'d to run in parallel
 - These smaller sub-tasks are join()'d, but a result is not returned directly
 - e.g., results may be stored in a collection, such as a List, or an array

```
abstract class RecursiveAction
    extends ForkJoinTask<Void> {
    protected abstract Void
        compute();
    ...
}
```



Key Methods in Java RecursiveAction

- RecursiveAction extends ForkJoinTask & does not return a result
 - Subclasses override compute() to perform task's main computation
 - The fork-join framework calls exec() to execute the task

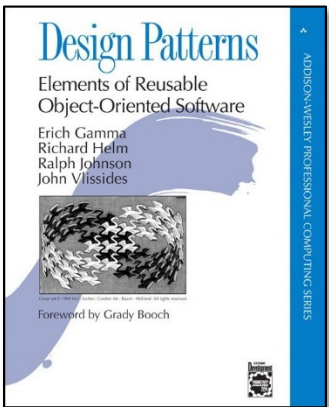
```
abstract class RecursiveAction
    extends ForkJoinTask<Void> {
    protected abstract Void
        compute();

    protected final boolean exec() {
        compute();
        return true;
    }
    ...
}
```

exec() is an abstract method in ForkJoinTask that's overridden in RecursiveAction

Key Methods in Java RecursiveAction

- RecursiveAction extends ForkJoinTask & does not return a result
 - Subclasses override compute() to perform task's main computation
 - The fork-join framework calls exec() to execute the task



```
abstract class RecursiveAction
    extends ForkJoinTask<Void> {
    protected abstract Void
        compute();

    protected final boolean exec() {
        compute();
        return true;
    }
    ...
}
```

exec() is a template method & compute() is a hook method

See en.wikipedia.org/wiki/Template_method_pattern

Key Methods in Java RecursiveAction

- RecursiveAction extends ForkJoinTask & does not return a result
 - Subclasses override compute() to perform task's main computation
 - The fork-join framework calls exec() to execute the task

```
abstract class RecursiveAction
    extends ForkJoinTask<Void> {
    protected abstract Void
        compute();

    protected final boolean exec() {
        compute();
        return true;
    }
    ...
}
```

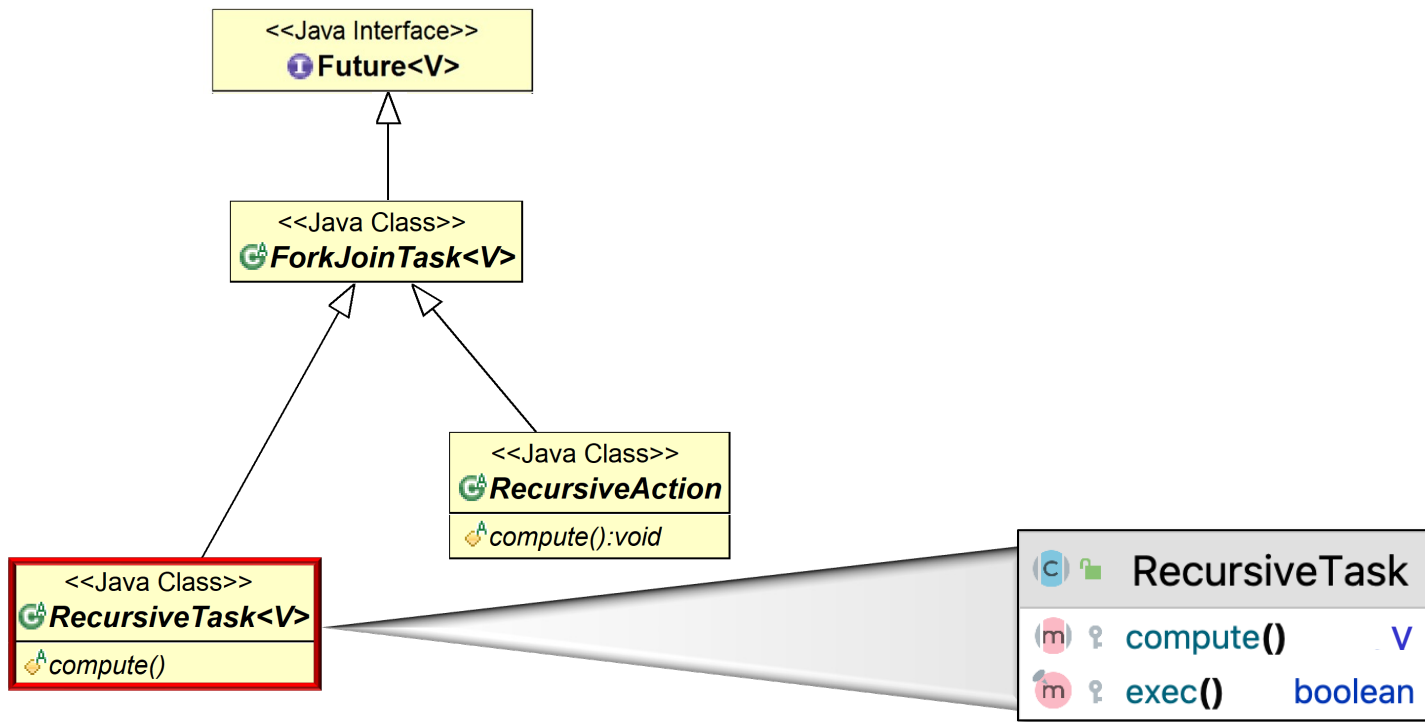
The result of compute() is not stored for subsequent access

Key Methods in the Java RecursiveTask

Key Methods in Java RecursiveTask

- RecursiveTask extends ForkJoinTask to return a result

```
abstract class RecursiveTask<V>  
    extends ForkJoinTask<V> {  
    ...  
}
```

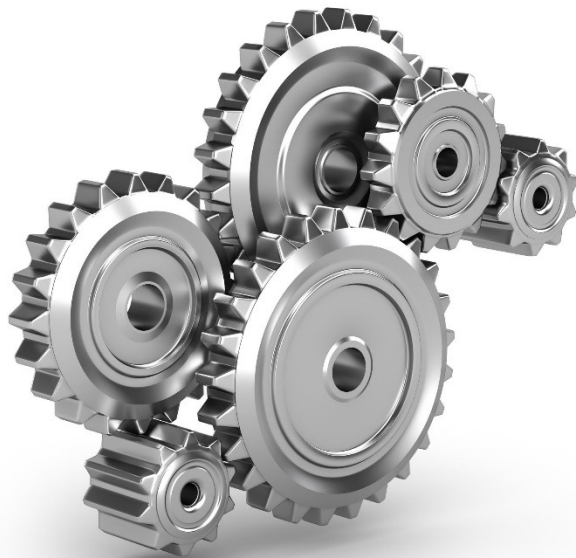


See docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html

Key Methods in Java RecursiveTask

- RecursiveTask extends ForkJoinTask to return a result
- Subclasses override compute() to perform task's main computation

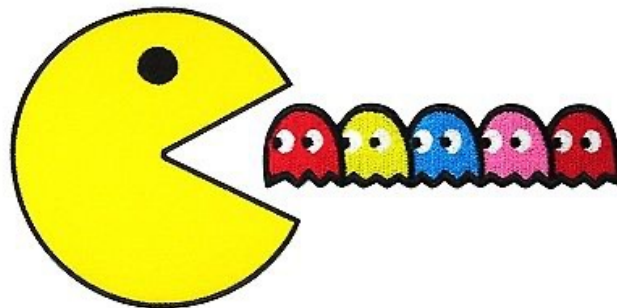
```
abstract class RecursiveTask<V>  
    extends ForkJoinTask<V> {  
    protected abstract V  
        compute ();  
    ...  
}
```



Key Methods in Java RecursiveTask

- RecursiveTask extends ForkJoinTask to return a result
- Subclasses override compute() to perform task's main computation
- If data size is below a certain threshold perform work directly

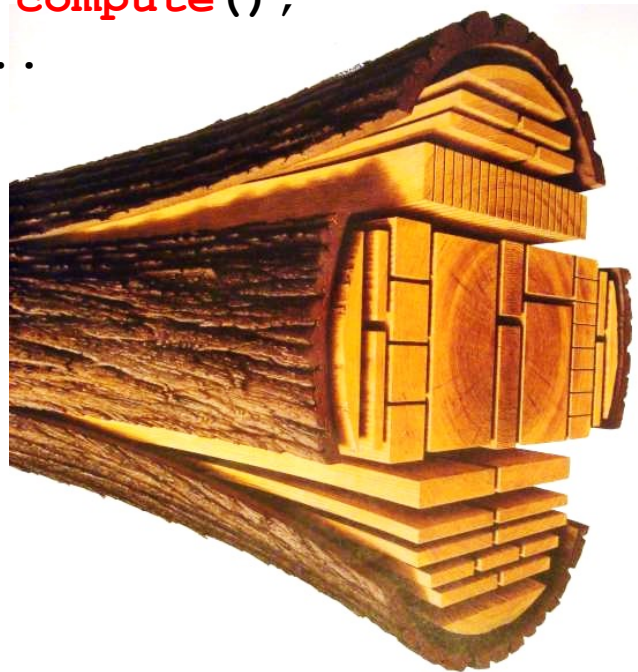
```
abstract class RecursiveTask<V>  
    extends ForkJoinTask<V> {  
    protected abstract V  
        compute ();  
    ...  
}
```



Key Methods in Java RecursiveTask

- RecursiveTask extends ForkJoinTask to return a result
- Subclasses override compute() to perform task's main computation
 - If data size is below a certain threshold perform work directly
 - If data size is large, split work into smaller sub-tasks that are fork()'d to run in parallel

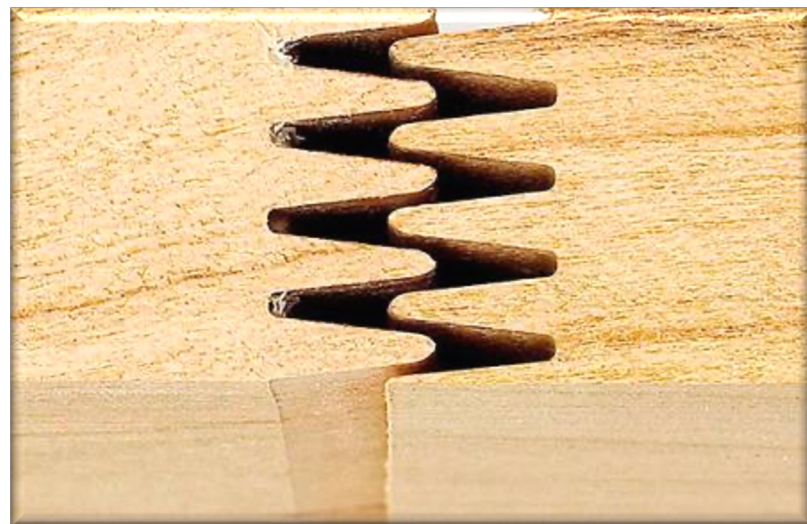
```
abstract class RecursiveTask<V>  
    extends ForkJoinTask<V> {  
    protected abstract V  
        compute ();  
    ...  
}
```



Key Methods in Java RecursiveTask

- RecursiveTask extends ForkJoinTask to return a result
 - Subclasses override compute() to perform task's main computation
 - If data size is below a certain threshold perform work directly
 - If data size is large, split work into smaller sub-tasks that are fork()'d to run in parallel
 - Results of these smaller sub-tasks are join()'d into a merged result

```
abstract class RecursiveTask<V>  
    extends ForkJoinTask<V> {  
    protected abstract V  
        compute ();  
    ...  
}
```



Key Methods in Java RecursiveTask

- RecursiveTask extends ForkJoinTask to return a result
 - Subclasses override compute() to perform task's main computation
 - The fork-join framework calls exec() to execute the task

```
abstract class RecursiveTask<V>
    extends ForkJoinTask<V> {
    protected abstract V
        compute();

    V result;

    protected final boolean exec() {
        result = compute();
        return true;
    }
    ...
}
```

exec() is an abstract (template) method in Fork JoinTask that's overridden in RecursiveTask

Key Methods in Java RecursiveTask

- RecursiveTask extends ForkJoinTask to return a result
 - Subclasses override compute() to perform task's main computation
 - The fork-join framework calls exec() to execute the task

```
abstract class RecursiveTask<V>
    extends ForkJoinTask<V> {
    protected abstract V
        compute();

    V result;

    protected final boolean exec() {
        result = compute();
        return true;
    }
    ...
}
```

The result of compute() is stored for subsequent access

End of Key Methods in the Java RecursiveAction & RecursiveTask Subclasses