The Java Fork-Join Pool: Key Methods in RecursiveAction & RecursiveTask

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 RecursiveAction & RecursiveTask classes
Key Methods in the Java RecursiveAction
Key Methods in Java RecursiveAction

- RecursiveAction extends ForkJoinTask & does not return a result

```java
abstract class RecursiveAction extends ForkJoinTask<Void> {
    
    // RecursiveAction methods
    void compute();
}
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveAction.html](https://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&lt;Void&gt; {
    protected abstract Void compute();
    ...
}

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html#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();
    ...
}
RecursiveAction extends ForkJoin Task & 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

```java
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 an array

abstract class RecursiveAction extends ForkJoinTask<Void> {
    protected abstract Void compute();
    ...
}
RecursiveAction extends ForkJoinTask & does not return a result

- Subclasses override compute() to perform task’s main computation
- The fork-join pool framework calls exec() to execute the task

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

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

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html#exec
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 pool framework calls exec() to execute the task

```java
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](en.wikipedia.org/wiki/Template_method_pattern)
RecursiveAction extends ForkJoinTask & does not return a result

- Subclasses override compute() to perform task’s main computation
- The fork-join pool 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

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

RecursiveTask<V> {
    compute()
}
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html](https://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();
    ...

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html#compute
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();
  ...
}
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

```java
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 pool 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;
  }

  ...

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/RecursiveTask.html#exec
RecursiveTask extends ForkJoinTask to return a result

- Subclasses override compute() to perform task’s main computation
- The fork-join pool framework calls exec() to execute the task

```java
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 the Java Fork-Join Pool: Key Methods in RecursiveAction & RecursiveTask