

# Java SE 8 Programming

Robotron Datenbank-Software GmbH  
Schulungszentrum  
Heilbronner Straße 21  
01189 Dresden



Java kann Aufgaben **effizient auf Kerne aufteilen**.

**Bitte wählen Sie eine Kategorie, um näheres zu erfahren!**



**Einleitung**



**Aufgabenstellung**



**Ergebnis**



**Weitere Informationen**



Mit Klick auf dieses Symbol  
gelangen Sie auf dieses Seite.

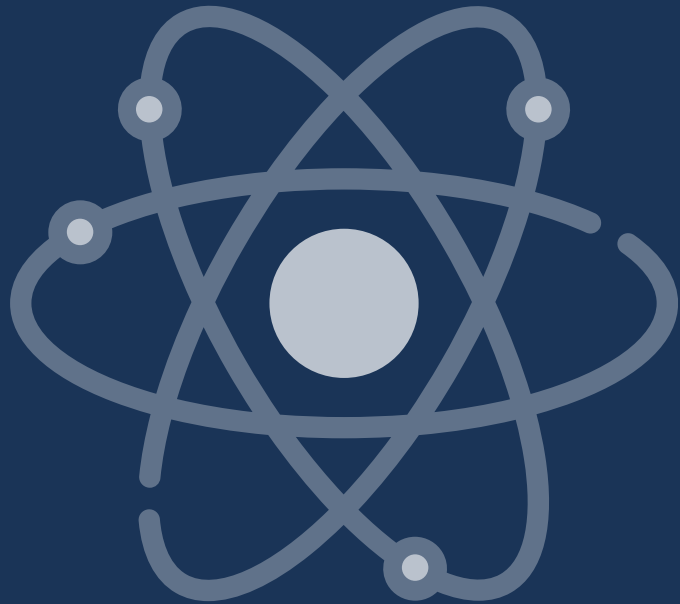


Mit diesen Symbolen  
navigieren Sie vor und zurück.



# KERNENERGIE?

**JA BITTE!**



Nutze ich die  
**Kraft meiner Kerne?**

Schließlich habe ich  
diese bezahlt!

**Java** ist eine der am weitesten verbreiteten Programmiersprachen. Es bietet komfortable Möglichkeiten zur Parallelisierung des Programmablaufes.

Anwendbar sind diese Features für **parallelisierbare Aufgaben**. Ein Beispiel ist die Bearbeitung großer, aufteilbarer Datenmengen.

Da hierbei lediglich die Verteilung der Aufgabe auf die Ressource Kerne unterstützt wird, sollte die Performance der Aufgabe hauptsächlich von dieser Ressource abhängen. Eine Aufgabe mit geringer CPU-Last und z.B. hoher Lese-Schreiblast kann nicht profitieren, evtl. sogar langsamer werden.

Die Suche eines Maximalwertes in einer größeren Datenmenge ist eine durch Aufteilung gut parallelisierbare Aufgabe.

Die Methode `calcIt(int i)` dient nur dazu, den Vorgang CPU-lastig zu machen.

Der Klick auf die Lupen zeigt den Code vergrößert.

A screenshot of the NetBeans IDE 8.0.2 interface. The main editor window displays the source code for `ForkJoinTest.java`. The code includes a `main` method that generates an array of random integers and then processes it using three different parallelization techniques: `SingleThreaded`, `ForkJoin`, and `Parallel Stream`. Each technique's code block is enclosed in a box with a magnifying glass icon in the bottom right corner, indicating that clicking the icon will zoom in on that specific code section. The `calcIt` method is also visible at the bottom of the code block. The IDE's Navigator and Output windows are also visible.



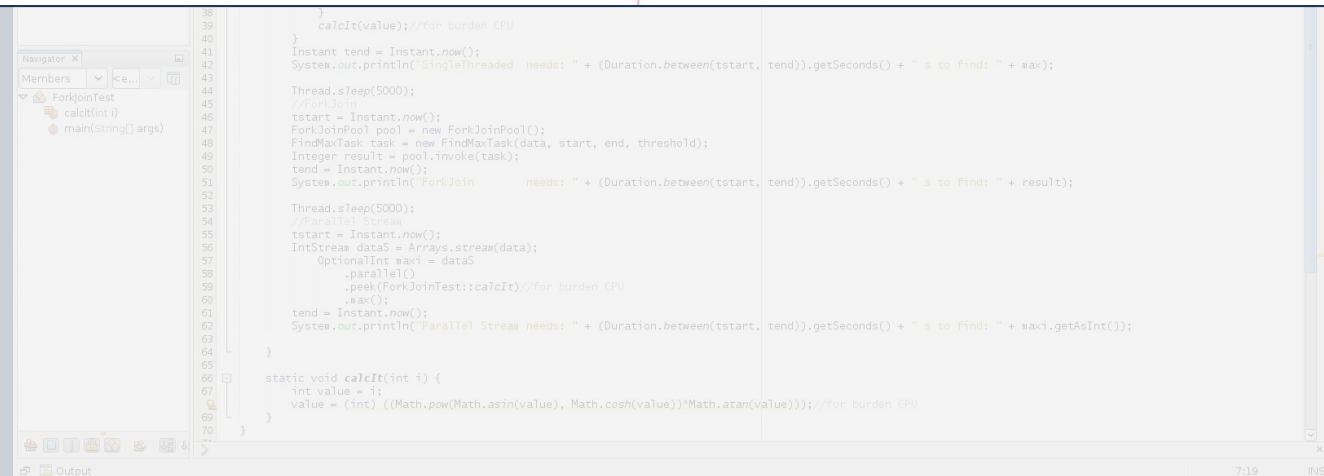
... zuerst **Seriell**  
(Single Threaded)

... nach den Code-  
abschnitten gibt es  
jeweils 5 Sekunden  
Erholung ...



```
//SingleThread
Instant tstart = Instant.now();
for (int value : data) {
    if (value > max) {
        max = value;
    }
    calcIt(value);//for burden CPU
}
Instant tend = Instant.now();
System.out.println("SingleThreaded needs: " + (Duration.between(tstart, tend)).getSeconds() + " s to find: " + max);

Thread.sleep(5000);
```



The screenshot shows the NetBeans IDE interface. The main editor window displays the following code:

```
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
}

static void calcIt(int i) {
    int value = i;
    value = (int) ((Math.pow(Math.asin(value), Math.cosh(value))*Math.atan(value)));//for burden CPU
}

}

Instant tend = Instant.now();
System.out.println("SingleThreaded needs: " + (Duration.between(tstart, tend)).getSeconds() + " s to find: " + max);

Thread.sleep(5000);
//ForkJoin
tstart = Instant.now();
ForkJoinPool pool = new ForkJoinPool();
FindMaxTask task = new FindMaxTask(data, start, end, threshold);
Integer result = pool.invoke(task);
tend = Instant.now();
System.out.println("ForkJoin needs: " + (Duration.between(tstart, tend)).getSeconds() + " s to find: " + result);

Thread.sleep(5000);
//Parallel Stream
tstart = Instant.now();
IntStream dataS = Arrays.stream(data);
OptionalInt maxI = dataS
    .parallel()
    .peek(ForkJoinTest::calcIt)//for burden CPU
    .max();
tend = Instant.now();
System.out.println("Parallel Stream needs: " + (Duration.between(tstart, tend)).getSeconds() + " s to find: " + maxI.getAsInt());

}
```

The Output window at the bottom shows the following output:

```
SingleThreaded needs: 5 s to find: 1000000000
ForkJoin needs: 5 s to find: 1000000000
Parallel Stream needs: 5 s to find: 1000000000
```



... jetzt parallel mittels  
ForkJoin Framework ...



```
ForkJoinTest - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Source Packages
  ForkJoinTest
    Source Packages
      forkjoinTest
        FindMaxTask.java
        ForkJoinTest.java
        forkjoinTestVerbose
        Libraries
Source History
public class ForkJoinTest {
    17 public static void main(String[] args) throws InterruptedException {
    18     //int[] data = new int[1024 * 1024 * 180]; // -Max1500k
    19     int[] data = new int[1024 * 1024 * 150]; // -Max1000k
    20     int[] data = new int[1024 * 1024];
    21
    22     for (int i = 0; i < data.length; i++) {
    23         data[i] = ThreadLocalRandom.current().nextInt();
    24     }
    25
    26     int proziCount = Runtime.getRuntime().availableProcessors();
    27     System.out.println("The system needs: " + proziCount + " processor" + ((proziCount > 1) ? "s" : ""));
    28     int threshold = data.length / proziCount;
    29     int start = 0;
    30     int end = data.length - 1;
    31     int max = Integer.MAX_VALUE;
    32
    33     //SingleThread
    34     Instant tstart = Instant.now();
    35     //for (int i = 0; i < data.length; i++) {
    36         //int value = data[i];
    37         //value = (int) ((Math.pow(Math.asin(value), Math.cosh(value))) * Math.atan(value)); //for burden CPU
    38     }
    39     Instant tend = Instant.now();
    40     System.out.println("Single Thread needs: " + (Duration.between(tstart, tend)).getSeconds() + " s to find: " + max);
    41 }
}
```

```
//ForkJoin
tstart = Instant.now();
ForkJoinPool pool = new ForkJoinPool();
FindMaxTask task = new FindMaxTask(data, start, end, threshold);
Integer result = pool.invoke(task);
tend = Instant.now();
System.out.println("ForkJoin needs: " + (Duration.between(tstart, tend)).getSeconds() + " s to find: " + result);
```

```
52 Thread.sleep(5000);
53 //parallel stream
54 tstart = Instant.now();
55 IntStream dataS = Arrays.stream(data);
56 OptionalInt maxI = dataS
57     .parallel()
58     .peek(ForkJoinTest::calcIt)//for burden CPU
59     .max();
60
61 tend = Instant.now();
62 System.out.println("Parallel Stream needs: " + (Duration.between(tstart, tend)).getSeconds() + " s to find: " + maxI.getAsInt());
63
64 }
65
66 static void calcIt(int i) {
67     int value = i;
68     value = (int) ((Math.pow(Math.asin(value), Math.cosh(value))) * Math.atan(value)); //for burden CPU
69 }
70 }
Output
7:19 INS
```



... und nun ebenfalls parallel  
mit der **Stream API** ...

```
public class ForkJoinTest {
    public static void main(String[] args) throws InterruptedException {
        //int[] data = new int[1024 * 1024 * 180]; // -Max1500k
        int[] data = new int[1024 * 1024 * 150]; // -Max1000k
        //int[] data = new int[1024 * 1024];
        for (int i = 0; i < data.length; i++) {
            data[i] = ThreadLocalRandom.current().nextInt();
        }
        int proziCount = Runtime.getRuntime().availableProcessors();
        System.out.println("The system needs: " + proziCount + " processor" + ((proziCount > 1) ? "s" : ""));
        int threshold = data.length / proziCount;
        int start = 0;
        int end = data.length - 1;
        int max = Integer.MIN_VALUE;
        //SingleThreaded
        Instant tstart = Instant.now();
        for (int value : data) {
            if (value > max) {
                max = value;
            }
            calcIt(value); //for burden CPU
        }
        Instant tend = Instant.now();
        System.out.println("SingleThreaded needs: " + (Duration.between(tstart, tend)).getSeconds() + " s to find: " + max);
        //ForkJoin
        Thread.sleep(5000);
        tstart = Instant.now();
        ForkJoinPool pool = new ForkJoinPool();
        FindMaxTask task = new FindMaxTask(data, start, end, threshold);
        Integer result = pool.invoke(task);
        tend = Instant.now();
        System.out.println("ForkJoin needs: " + (Duration.between(tstart, tend)).getSeconds() + " s to find: " + result);
    }
}
```

```
//Parallel Stream
tstart = Instant.now();
IntStream dataS = Arrays.stream(data);
    OptionalInt maxi = dataS
        .parallel()
        .peek(ForkJoinTest::calcIt)//for burden CPU
        .max();
tend = Instant.now();
System.out.println("Parallel Stream needs: " + (Duration.between(tstart, tend)).getSeconds() + " s to find: " + maxi.getAsInt());
```

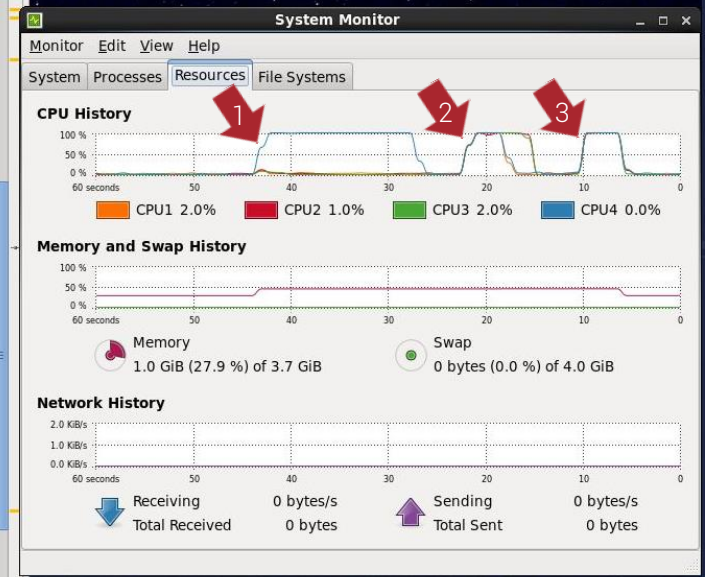


Und wie sieht das der Rechner?

```
ForkJoinTest - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+F)

FindMaxTask.java x ForkJoinTest.java x
Source History
29 int start = 0;
30 int end = data.length - 1;
31 int max = Integer.MIN_VALUE;
32
33 //SingleThreaded
34 Instant tstart = Instant.now();
35 for (int value : data) {
36     if (value > max) {
37         max = value;
38     }
39     calcIt(value); //for burden CPU
40 }
41 Instant tend = Instant.now();
42 System.out.println("SingleThreaded needs: " + (Duration.between(tstart, tend)).getSeconds() + " s to find: " + max);
43
44 Thread.sleep(5000);
45 //ForkJoin
46 tstart = Instant.now();
47 ForkJoinPool pool = new ForkJoinPool();
48 FindMaxTask task = new FindMaxTask(data, start, end, threshold);
49 Integer result = pool.invoke(task);
50 tend = Instant.now();
51 System.out.println("ForkJoin needs: " + (Duration.between(tstart, tend)).getSeconds() + " s to find: " + result);
52
53 Thread.sleep(5000);
54 //Parallel Stream
55 tstart = Instant.now();
56 IntStream dataS = Arrays.stream(data);
57 OptionalInt maxi = dataS
58     .parallel()
59     .peek(ForkJoinTest::calcIt) //for burden CPU
60     .max();
61 tend = Instant.now();
62 System.out.println("Parallel Stream needs: " + (Duration.between(tstart, tend)).getSeconds() + " s to find: " + maxi.getAsInt());
63
64 }
65
66 static void calcIt(int i) {
67     int value = i;
68     value = (int) ((Math.pow(Math.asin(value), Math.cosh(value)) * Math.atan(value))); //for burden CPU
69 }
70 }

Output - ForkJoinTest (run) x
run:
The system owns 4 processors
SingleThreaded needs: 15 s to find: 2147483630
ForkJoin needs: 6 s to find: 2147483630
Parallel Stream needs: 4 s to find: 2147483630
BUILD SUCCESSFUL (total time: 36 seconds)
```



```
run:
The system owns 4 processors
SingleThreaded needs: 15 s to find: 2147483630
ForkJoin needs: 6 s to find: 2147483630
Parallel Stream needs: 4 s to find: 2147483630
BUILD SUCCESSFUL (total time: 36 seconds)
```



Besuchen Sie uns!



Wir bieten Ihnen viele weitere Kurse an und passend zum Thema folgende:

[Java SE 8 Fundamentals](#)

[Java EE 7: Front-end Web Application Development](#)

Wir freuen uns auf Sie in unserem modern ausgestatteten Schulungszentrum!

Kontakt:

Michaela Eisold, Schulungsassistentin

Telefon: +49 351 25859-2660

E-Mail: [schulung@robotron.de](mailto:schulung@robotron.de)

Internet: [www.robotron.de](http://www.robotron.de)

