

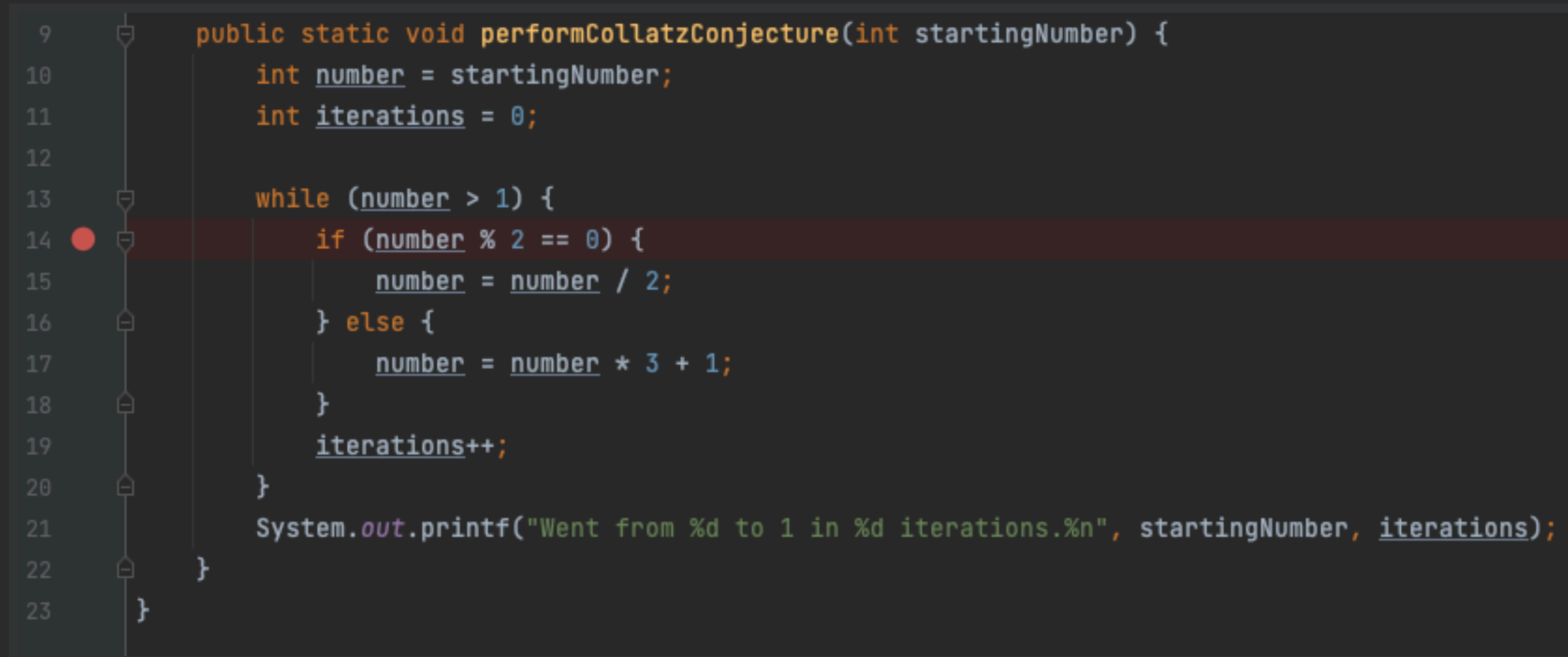
# DEBUGGING IN INTELLIJ

---



## EINEN BREAKPOINT ERSTELLEN

- Im linken Bereich klicken um Breakpoint zu erstellen

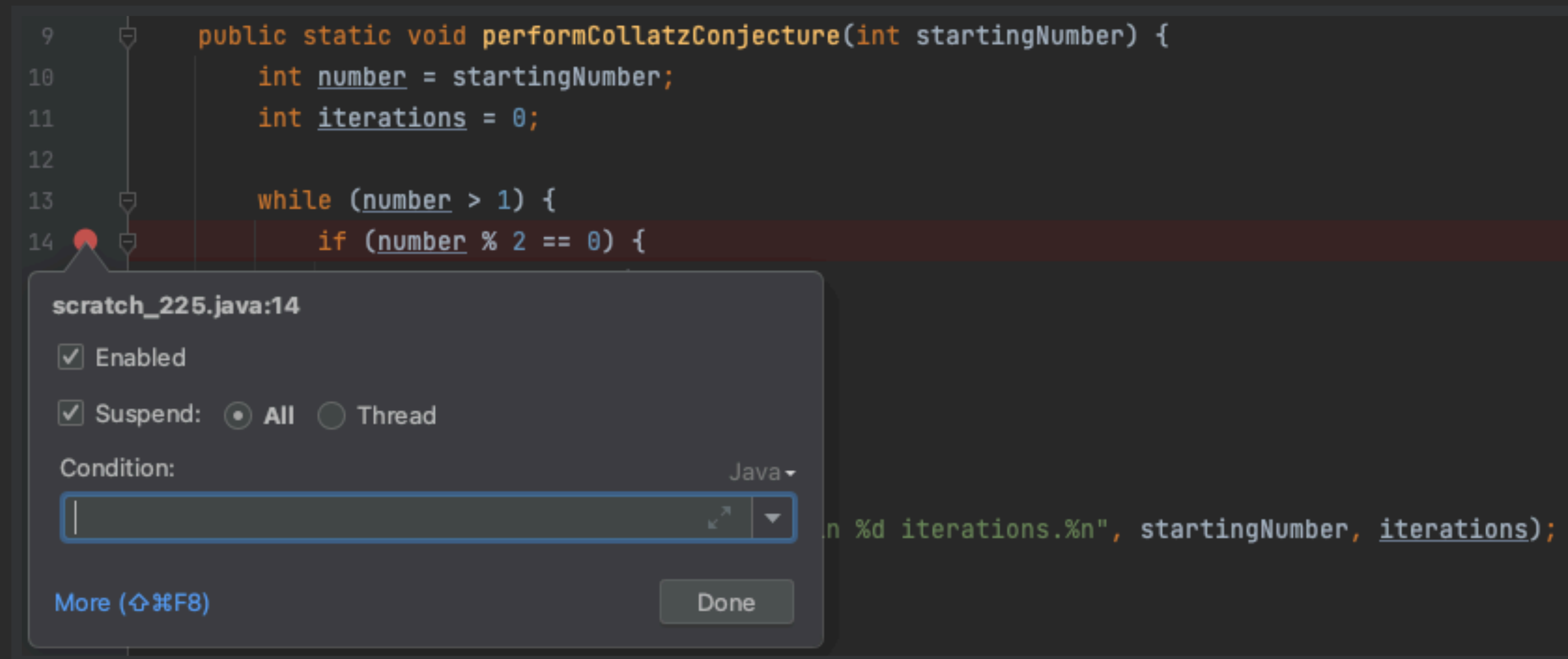


The screenshot shows a code editor with a Java method `performCollatzConjecture`. A red dot, representing a breakpoint, is placed on the left margin next to line 14, which contains the `if` statement `if (number % 2 == 0) {`. The code is as follows:

```
9      public static void performCollatzConjecture(int startingNumber) {
10          int number = startingNumber;
11          int iterations = 0;
12
13          while (number > 1) {
14              if (number % 2 == 0) {
15                  number = number / 2;
16              } else {
17                  number = number * 3 + 1;
18              }
19              iterations++;
20          }
21          System.out.printf("Went from %d to 1 in %d iterations.%n", startingNumber, iterations);
22      }
23  }
```

## EINEN BREAKPOINT BEARBEITEN

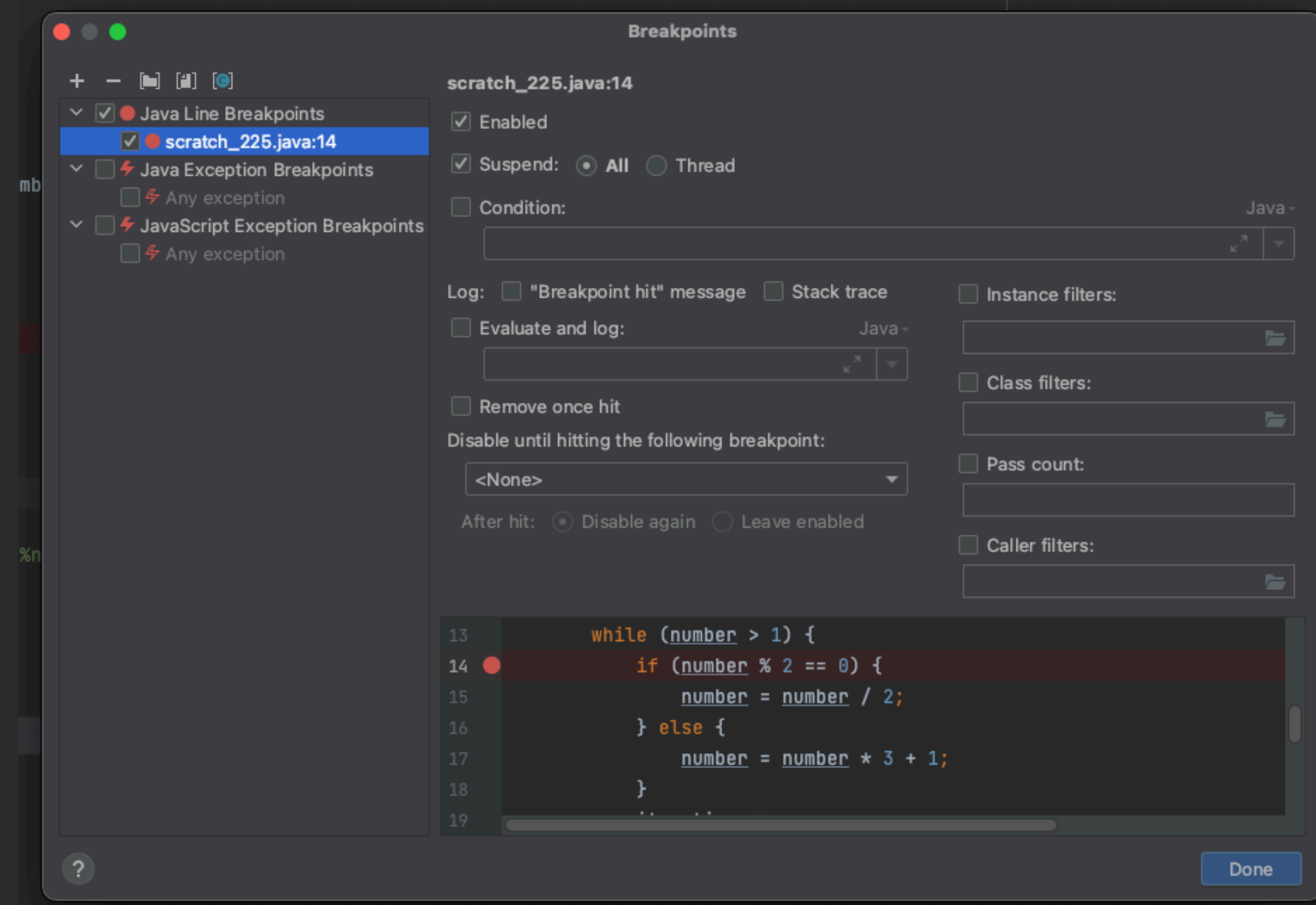
- ▶ Mit der rechten Maustaste auf Breakpoint klicken zum Bearbeiten



# EINEN BREAKPOINT BEARBEITEN

## Interessante Optionen:

- ▶ **Suspend:**  
Steuert, ob und „wie viel“ beim Breakpoint angehalten wird
- ▶ **Condition:**  
Nur wenn Ausdruck erfüllt ist, wird beim Breakpoint gehalten
- ▶ **Evaluate and log:**  
Der angegebene Ausdruck wird ausgewertet und in der Konsole ausgegeben



## WENN EIN BREAKPOINT ERREICHT WIRD

- ▶ Programm im Debug-Mode starten
- ▶ Blaue Zeile zeigt an, dass Programm dort stillsteht

```
9      public static void performCollatzConjecture(int startingNumber) {  startingNumber: 7
10          int number = startingNumber;  startingNumber: 7    number: 13
11          int iterations = 0;  iterations: 7
12
13          while (number > 1) {
14  ✓      if (number % 2 == 0 = false) {  number: 13
15              number = number / 2;
16          } else {
17              number = number * 3 + 1;
18          }
19          iterations++;
20      }
21      System.out.printf("Went from %d to 1 in %d iterations.%n", startingNumber, iterations);
22  }
23  }
```

## WENN EIN BREAKPOINT ERREICHT WIRD


 Step Over: Springe zur nächsten Zeile

 Step Into: Springe in Methode


 Evaluate Expression: Führe Code „live“ aus.


 Step Out: Springe aus Methode

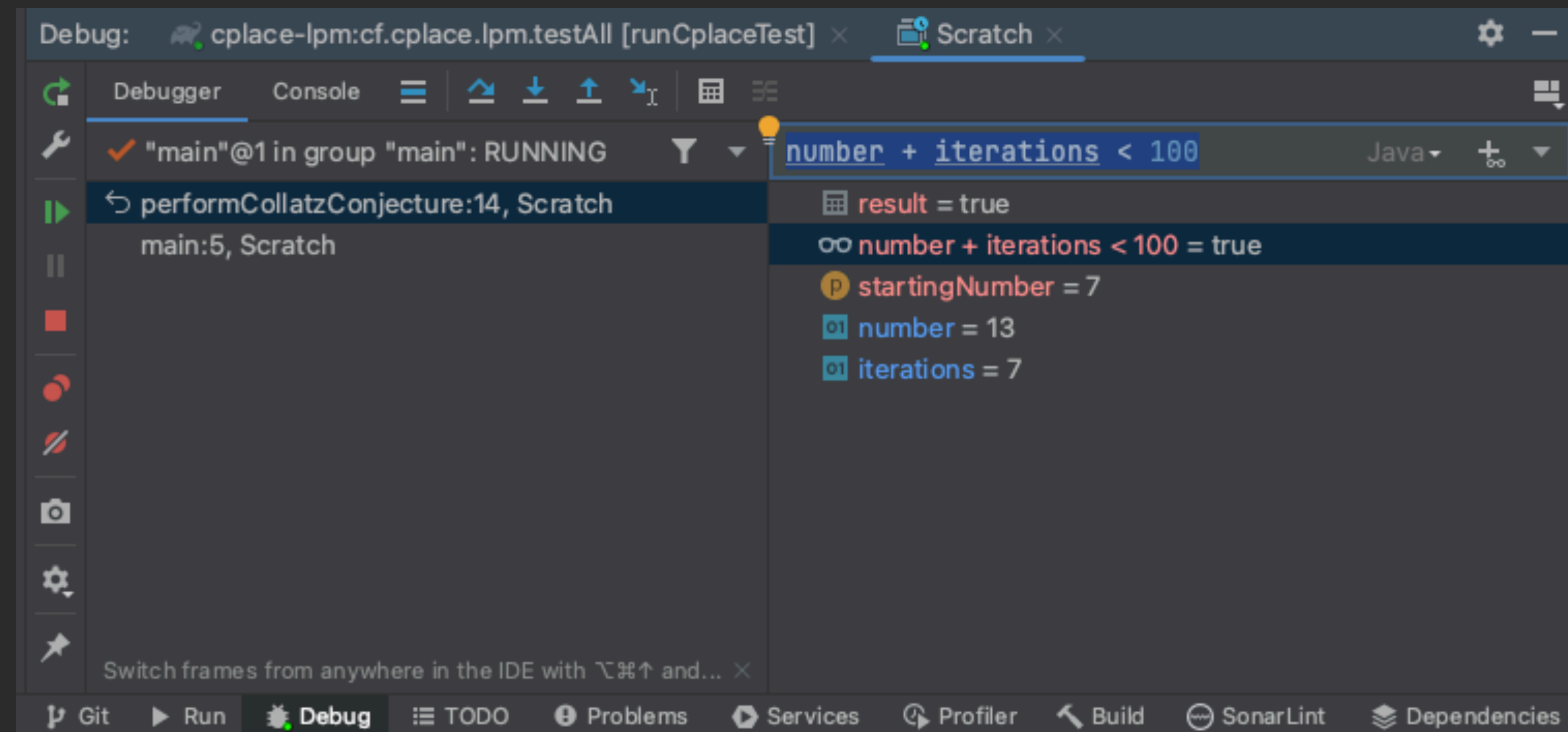
▶ Linke Seite: Stacktrace

 Resume: Fahre bis zum nächsten Breakpoint fort

▶ Rechte Seite: Aktuelle Felder / Watches

 View Breakpoints...:  
Öffne Dialog zum Bearbeiten  
von Breakpoints

 Mute Breakpoints:  
Überspringe temporär alle  
Breakpoints





# STACKTRACE

Aufrufreihenfolge

```
3 public static void main(String[] args) {  
4     firstMethod();  
5 }  
6  
7 private static void firstMethod() {  
8     secondMethod();  
9 }  
10  
11 private static void secondMethod() {  
12     thirdMethod();  
13 }  
14  
15 private static void thirdMethod() {  
16     throw new IllegalStateException("This is an example!");  
17 }
```

The Debugger window shows the call stack for the current thread. The top frame is highlighted with a blue arrow, indicating the current execution point.

Frame
thirdMethod:16, Scratch
secondMethod:12, Scratch
firstMethod:8, Scratch
main:4, Scratch

Switch frames from anywhere in the IDE with `⌘⇧↑` and `⌘⇧↓`.

The Console window displays the output of the program, including the exception message and the stack trace.

```
/Users/fheerdelokal/Library/Java/JavaVirtualMachines/corretto-17.0.6/Contents/Home/bin/java ...  
Connected to the target VM, address: '127.0.0.1:56776', transport: 'socket'  
Exception in thread "main" java.lang.IllegalStateException: This is an example!  
    at Scratch.thirdMethod(scratch_226.java:16)  
    at Scratch.secondMethod(scratch_226.java:12)  
    at Scratch.firstMethod(scratch_226.java:8)  
    at Scratch.main(scratch_226.java:4)  
Disconnected from the target VM, address: '127.0.0.1:56776', transport: 'socket'  
  
Process finished with exit code 1
```