

Oliver Geisel

Fakultät Informatik, Professur Softwaretechnologie

Debugging mit IDEs

Einführung ins Debuggen von Code und das Finden von Fehlern

8. Juni 2023

Inhalt

1. Wie es nicht geht!
2. Grundlagen
3. Der Debugger und seine Bestandteile in einer IDE
4. Debuggen mit Tests
5. Das Beispiel Bibliothek

Ziele

Kennen der Debugging-Grundlagen

Debugging mit der IDE

Kennen der Test-Grundlagen

Test mit IDEs

Bugs durch Tests finden

Motivation für Java-Features im Selbststudium

Was wird benötigt

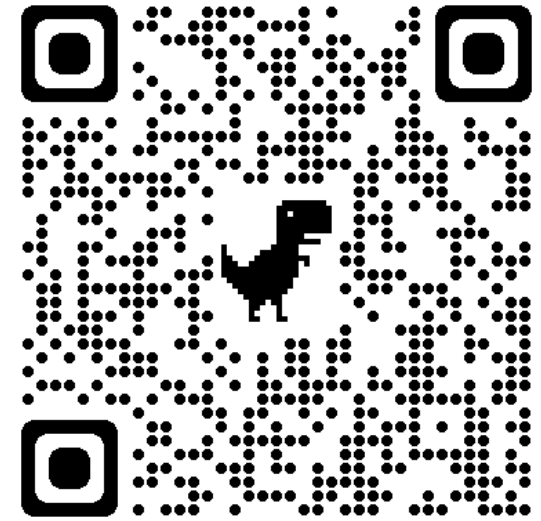
Ab Java 8 für Einführungsbeispiele

Ab Java 19 für das große Beispiel

IDE der Wahl

Das Repo: <https://github.com/OliverGeisel/Debug-Tutorial>

Maven



Bugs

Ein Bug ist ein (Laufzeit-)Fehler eines Programmes

Ursachen eines Bugs:

- Syntax
- Semantisch
- Logisch

Zeigen sich im „besten Fall“ als:

- Ein falsches Ergebnis
- Ein Absturz

Im schlimmsten Fall unentdeckt



Finde den Fehler!

```
public static int summeVonBis(int a, int b) {  
    if (a > b) {  
        int temp = a;  
        a = b;  
        b = temp;  
    }  
    int summe = 0;  
    for (int run = a; run < b; run++) {  
        summe += run;  
    }  
    return summe;  
}
```

```
public static void main(String[] args) {  
  
    // Gib die Summe der Zahlen von a bis b (inklusive a,b) aus!  
    // es können auch negative Zahlen eingegeben werden  
    int a = 2;  
    int b = 7;  
    int c = summeVonBis(a, b);  
  
    System.out.print("Summe der Zahlen a bis b = " + c); // Hier ist es 20  
}
```

Ergebnis ist 20

Muss aber 27 sein

System.out.print()

Eingrenzen eines Fehlers durch setzen von System.out.print(variable) an einer Stelle im Code Probleme:

- Nicht unterbrechbar
- Muss weitere Informationen mit liefern
- Bedingungen/ Filterung muss programmiert werden
- Keine Veränderung

```
Summe ist: 2  
run ist: 3  
Summe ist: 5  
run ist: 4  
Summe ist: 9  
run ist: 5  
Summe ist: 14  
run ist: 6  
Summe ist: 20  
Summe am Ende der Methode: 20  
Summe der Zahlen a bis b = 20  
Process finished with exit code 0
```

```
if (a > b) {  
    int temp = a;  
    a = b;  
    b = temp;  
}  
int summe = 0;  
for (int run = a; run < b; run++) {  
    System.out.println("run ist: " + run);  
    summe += run;  
    System.out.println("Summe ist: " + summe);  
}  
System.out.println("Summe am Ende der Methode: " + summe);  
return summe;
```

Logging ist nicht viel besser - Aber eine Unterstützung

Erzeugen eines Logger-Objektes

Kategorisierung durch Log-Level

Probleme:

- Nicht unterbrechbar
- Keine Veränderung

```
May 05, 2022 11:48:03 PM BreakpointLogging summeVonBis
INFO: run ist: 5
May 05, 2022 11:48:03 PM BreakpointLogging summeVonBis
INFO: Summe ist: 14
May 05, 2022 11:48:03 PM BreakpointLogging summeVonBis
INFO: run ist: 6
May 05, 2022 11:48:03 PM BreakpointLogging summeVonBis
INFO: Summe ist: 20
May 05, 2022 11:48:03 PM BreakpointLogging summeVonBis
INFO: Summe am Ende der Methode: 20
Summe der Zahlen a bis b = 20
Process finished with exit code 0
```

```
private static Logger log = Logger.getLogger(name: "Debug-Logger");
1 usage
public static int summeVonBis(int a, int b) {
    if (a > b) {
        int temp = a;
        a = b;
        b = temp;
    }
    int summe = 0;
    for (int run = a; run < b; run++) {
        log.info(msg: "run ist: " + run);
        summe += run;
        log.info(msg: "Summe ist: " + summe);
    }
    log.info(msg: "Summe am Ende der Methode: " + summe);
    return summe;
}
```


Stack Trace - Den Ort des Fehlers eingrenzen

Informiert, dass eine nicht gefangene Exception auftrat -> Programm bricht ab

Enthält:

- Threadname
- Typ der Exception
- Fehlermeldung/Information
- Trace

```
Exception in thread "main" java.lang.IllegalArgumentException Create breakpoint : java.util.NoSuchElementException: No value present
    at ExceptionBeispiele.gemeineException(ExceptionBeispiele.java:35)
    at ExceptionBeispiele.main(ExceptionBeispiele.java:71)
Caused by: java.util.NoSuchElementException Create breakpoint : No value present
    at java.base/java.util.Optional.get(Optional.java:143)
    at ExceptionBeispiele.tiefeException(ExceptionBeispiele.java:24)
    at java.base/java.util.Optional.orElseGet(Optional.java:364)
    at ExceptionBeispiele.arbeiten(ExceptionBeispiele.java:48)
    at ExceptionBeispiele.gemeineException(ExceptionBeispiele.java:32)
    ... 1 more
```

Stack Trace – Beispiel 1

Wo wurde die Exception geworfen?
In Zeile 31

```
Exception in thread "main" java.lang.NullPointerException: Cannot invoke "String.toUpperCase()" because "ExceptionBeispiele.hallo" is null
    at ExceptionBeispiele.NullPointerBeispiel(ExceptionBeispiele.java:31)
    at ExceptionBeispiele.main(ExceptionBeispiele.java:78)
```

```
Process finished with exit code 1
```

Stack Trace – Beispiel 2

```
Exception in thread "main" java.lang.RuntimeException: Zur Laufzeit gab es einen Fehler!  
    at ExceptionBeispiele.einfacheException(ExceptionBeispiele.java:35)  
    at ExceptionBeispiele.main(ExceptionBeispiele.java:81)
```

```
Process finished with exit code 1
```

Stack Trace – Beispiel 3

```
Exception in thread "main" java.util.NoSuchElementException Create breakpoint : No value present
    at java.base/java.util.Optional.get(Optional.java:143)
    at ExceptionBeispiele.tiefeException(ExceptionBeispiele.java:40)
    at ExceptionBeispiele.main(ExceptionBeispiele.java:84)

Process finished with exit code 1
```

Stack Trace – Beispiel 4

Wo wurde eine Exception geworfen?

In Zeile 51 und 143

```
Exception in thread "main" java.lang.IllegalArgumentException Create breakpoint : java.util.NoSuchElementException: No value present
    at ExceptionBeispiele.gemeineException(ExceptionBeispiele.java:51)
    at ExceptionBeispiele.main(ExceptionBeispiele.java:87)
Caused by: java.util.NoSuchElementException Create breakpoint : No value present
    at java.base/java.util.Optional.get(Optional.java:143)
    at ExceptionBeispiele.tiefeException(ExceptionBeispiele.java:40)
    at java.base/java.util.Optional.orElseGet(Optional.java:364)
    at ExceptionBeispiele.arbeiten(ExceptionBeispiele.java:64)
    at ExceptionBeispiele.gemeineException(ExceptionBeispiele.java:48)
    ... 1 more
```

Der Debug-Modus

Startet Java-Anwendung innerhalb einer Debug-Umgebung

Debugger kontrolliert/ überwacht die Anwendung

Liefert detaillierte Informationen in speziellen Fenstern

Steuerung des Ablaufes

Start mit extra Knopf



Kontrollelemente

Project

Pull Requests

Files

core

person.staff

structure

BibliothekTest 29

LeseraumTest 29

ArbeitsplatzTest 2

RegalTest 29/04/20

BreakpointTest 29/04/20

TestErgebnisse 30/04/20

start

Breakpoint 01/05/2022 00:03, 53

ExceptionBeispiele 30/04/2022

FrameScope 29/04/2022 14:00, 3

target

Debug-Tutorial.iml 29/04/2022 12:37,

aufgabe.md 04/05/2022 10:59, 5,54 kB

pom.xml 30/04/2022 23:29, 1,31 kB 30/

External Libraries

Scratches and Consoles

Dateien

Oliver Geisel

```
public class Breakpoint {  
    5 usages  
    public static int summeVonBis(int a, int b) {  
        if (a > b = false) {  
            int temp = a;  
            a = b;  
            b = temp;  
        }  
        int summe = 0;  
        for (int run = a; run < b; run++) {  
            summe += run;  
        }  
        return summe;  
    }  
    public static void main(String[] args) {  
        // Gib die Summe der Zahlen von a bis b (inklusive a,b) aus!  
        // es können auch negative Zahlen eingegeben werden  
    }  
}
```

Editor

Frames

Threads

"main"@1 in group "main": RUNNING

summeVonBis:6, Breakpoint

main:24, Breakpoint

Switch frames from anywhere in the IDE with Strg+Alt+Ob...

Frames

a = 2

b = 7

Variablen

Overhead Memory

| Class | Count | Diff |
|-----------------------------------|-------|------|
| java.lang.module.ModuleDescript | 370 | 0 |
| java.util.HashMap | 327 | 0 |
| java.util.HashMap\$Node[] | 324 | 0 |
| java.util.HashSet | 262 | 0 |
| java.lang.Integer | 262 | 0 |
| java.util.ImmutableCollections\$S | 226 | 0 |
| java.lang.module.ModuleDescript | 168 | 0 |

Memory

Der Breakpoint

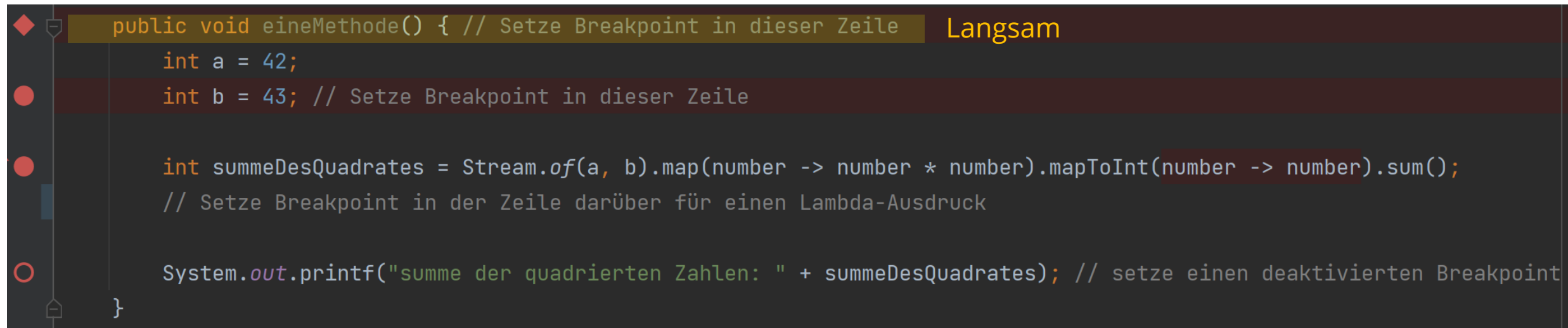
Eine Markierung in einer Zeile, die im Debug-Modus, die Ausführung anhält

Normalfall: an der linken Seite wird ein Punkt gesetzt

Spezialfall: Lambda, Methoden, Verschachtelung in Methoden- hängt von IDE ab

Sind deaktivierbar oder entfernbar

1. Methode
2. Zeile
3. Lambda
4. Deaktiviert



```
public void eineMethode() { // Setze Breakpoint in dieser Zeile Langsam
    int a = 42;
    int b = 43; // Setze Breakpoint in dieser Zeile

    int summeDesQuadrates = Stream.of(a, b).map(number -> number * number).mapToInt(number -> number).sum();
    // Setze Breakpoint in der Zeile darüber für einen Lambda-Ausdruck

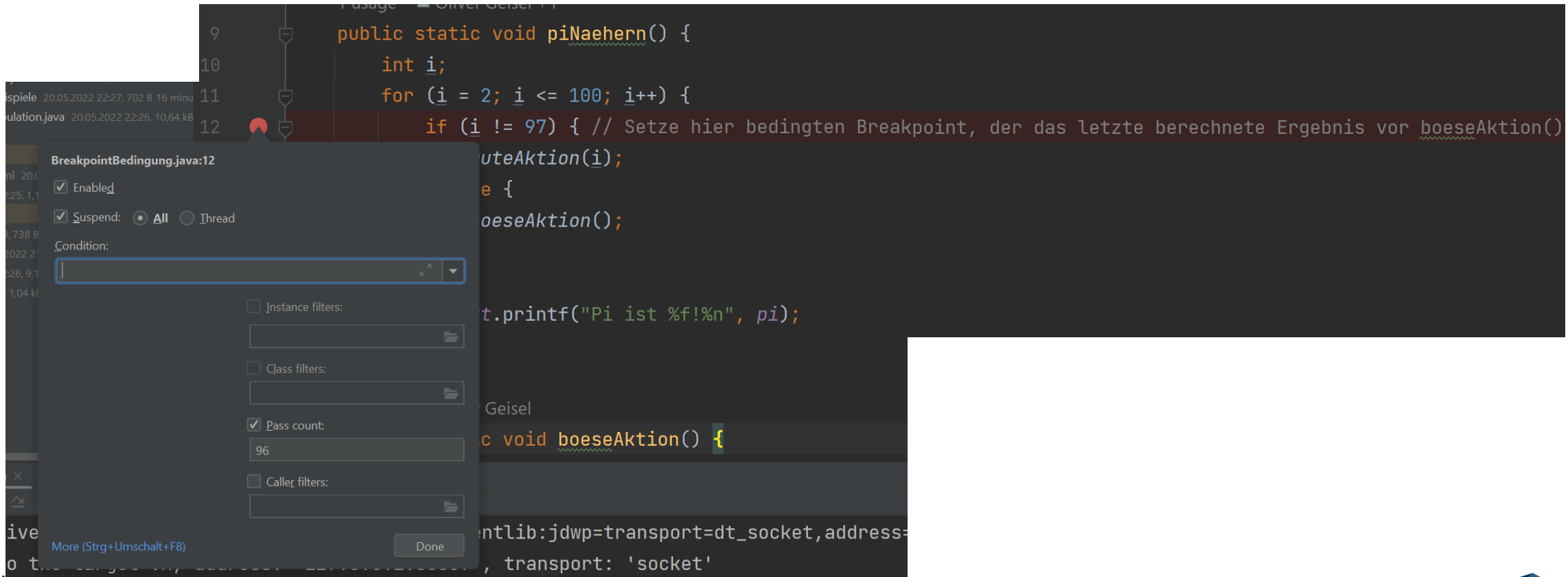
    System.out.printf("summe der quadrierten Zahlen: " + summeDesQuadrates); // setze einen deaktivierten Breakpoint
}
```


Bedingte Breakpoints

Breakpoints können auch halten wenn:

- Eine Expression true ist
- Eine Schleife n Mal durchlaufen ist

```
Pi ist am Anfang: 3.14072
Achtung! Der Prozess ist sehr rechenintensiv!
Abbruch mit Ctrl + C!
Pi ist 3,000000!
Benötigte Zeit: 16,774375800 sec.
```



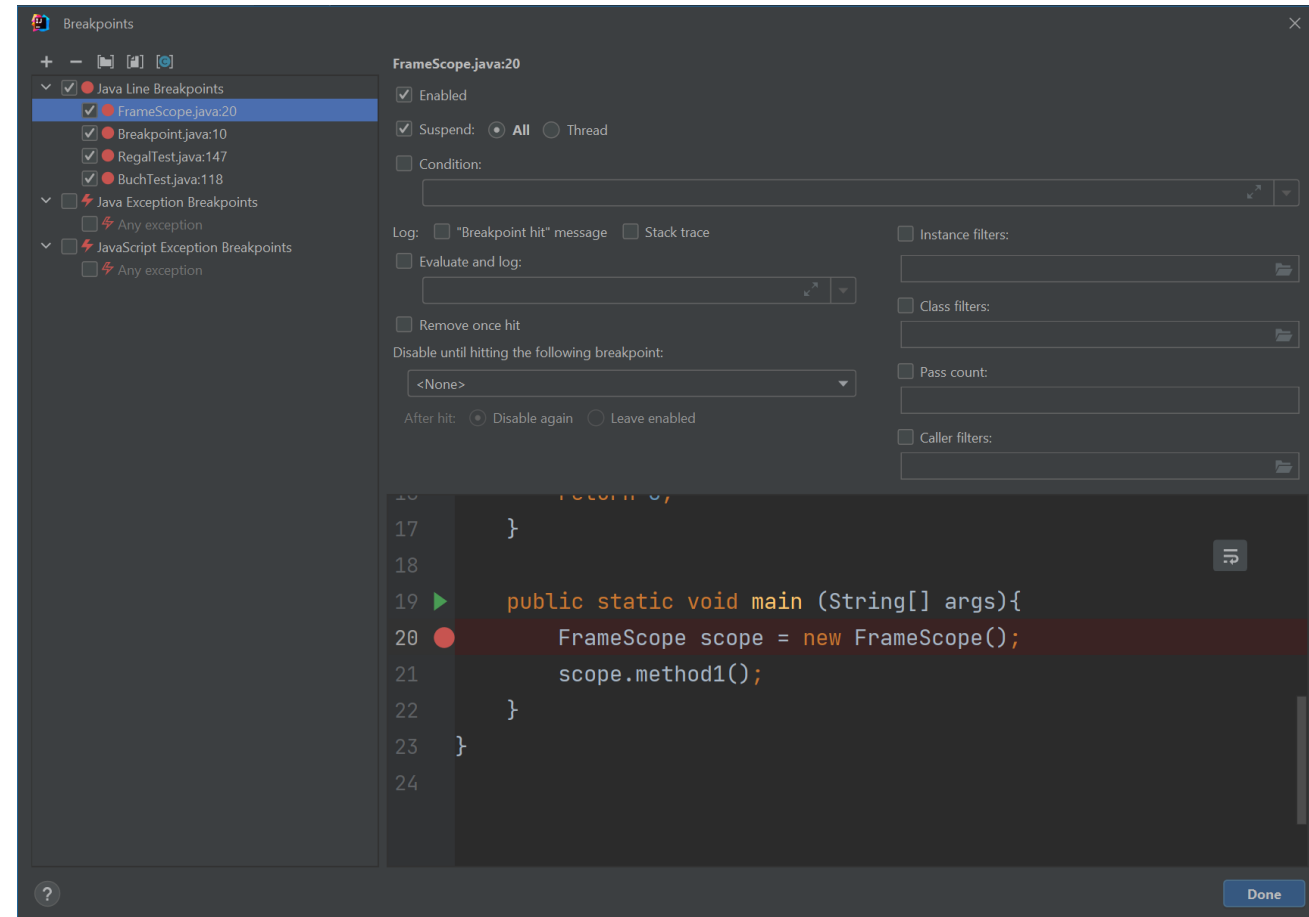
Breakpoint-Fenster

Zeigt alle gesetzten Breakpoints

Erlaubt diese zu aktivieren/ deaktivieren
oder auch wieder zu entfernen

Je nach IDE können Bedingungen hinzugefügt
werden

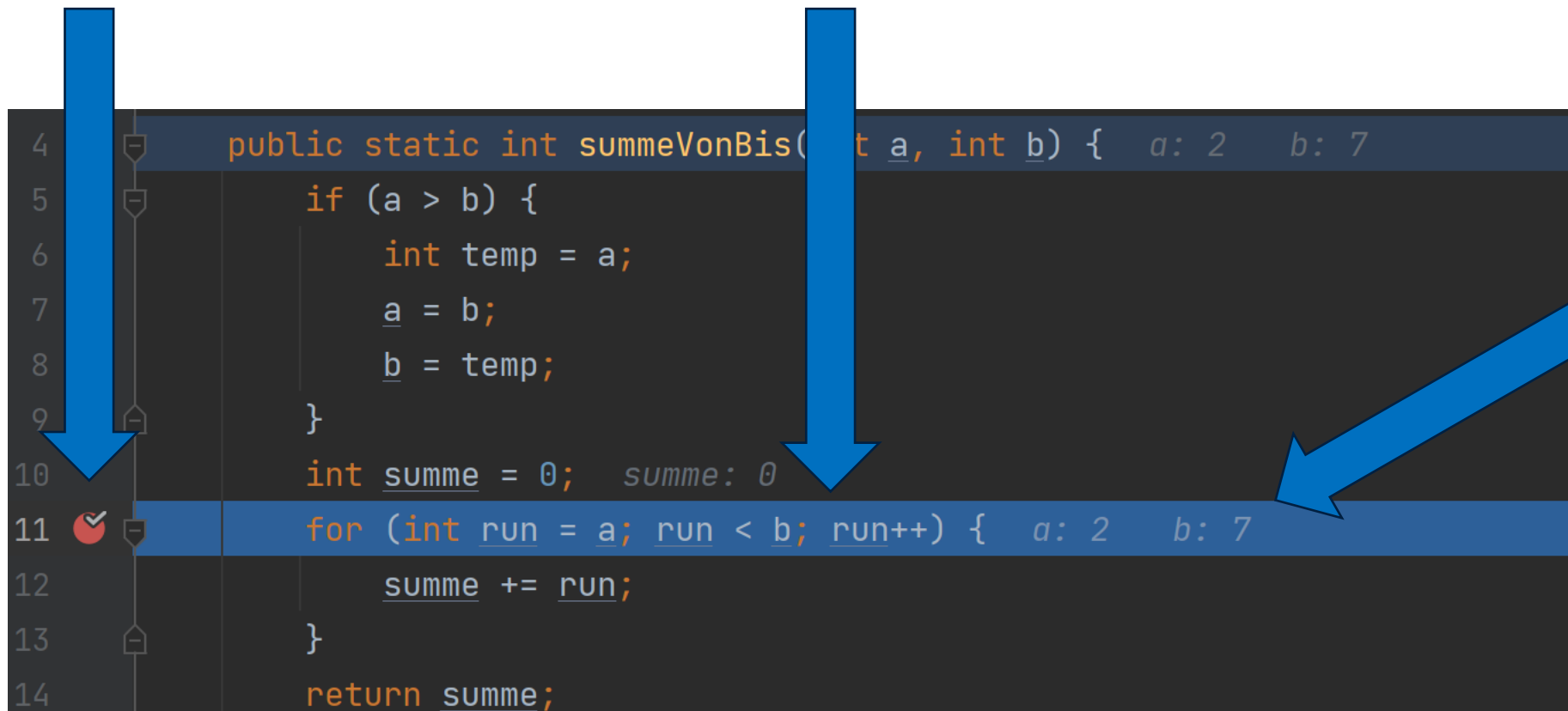
Logging ist möglich



Das Programmfenster

Breakpoint in Zeile 11

Blaue Zeile markiert, wo das Programm aktuell ist
Zeile wurde noch nicht ausgeführt!



```
4 public static int summeVonBis(int a, int b) { a: 2 b: 7
5     if (a > b) {
6         int temp = a;
7         a = b;
8         b = temp;
9     }
10    int summe = 0; summe: 0
11    for (int run = a; run < b; run++) { a: 2 b: 7
12        summe += run;
13    }
14    return summe;
```

Extra Informationen
zur Laufzeit

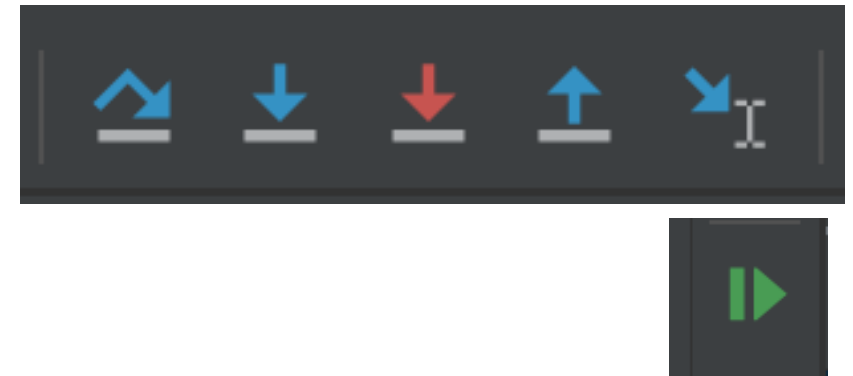


Springt zur
aktuellen Zeile

Schritt für Schritt zum Fehler

Es gibt verschiedene Varianten um das Programm weiter auszuführen

1. Step Over – führt nächsten Befehl aus, geht nicht in die Methode hinein
2. Step Into – geht in die nächste Methode hinein
3. Step Out – geht eine „Ebene“ höher
4. Resume – Programm läuft weiter
 - Bis zum nächsten Breakpoint
 - Bis zur Cursor-Position



Variablen-Fenster

Enthält alle Variablen im aktuellem Frame

Man kann auch weitere Variablen überwachen; werden immer aufgelistet

Alle Variablen können angesehen und geändert werden

Manche Watches werden von der IDE hinzugefügt

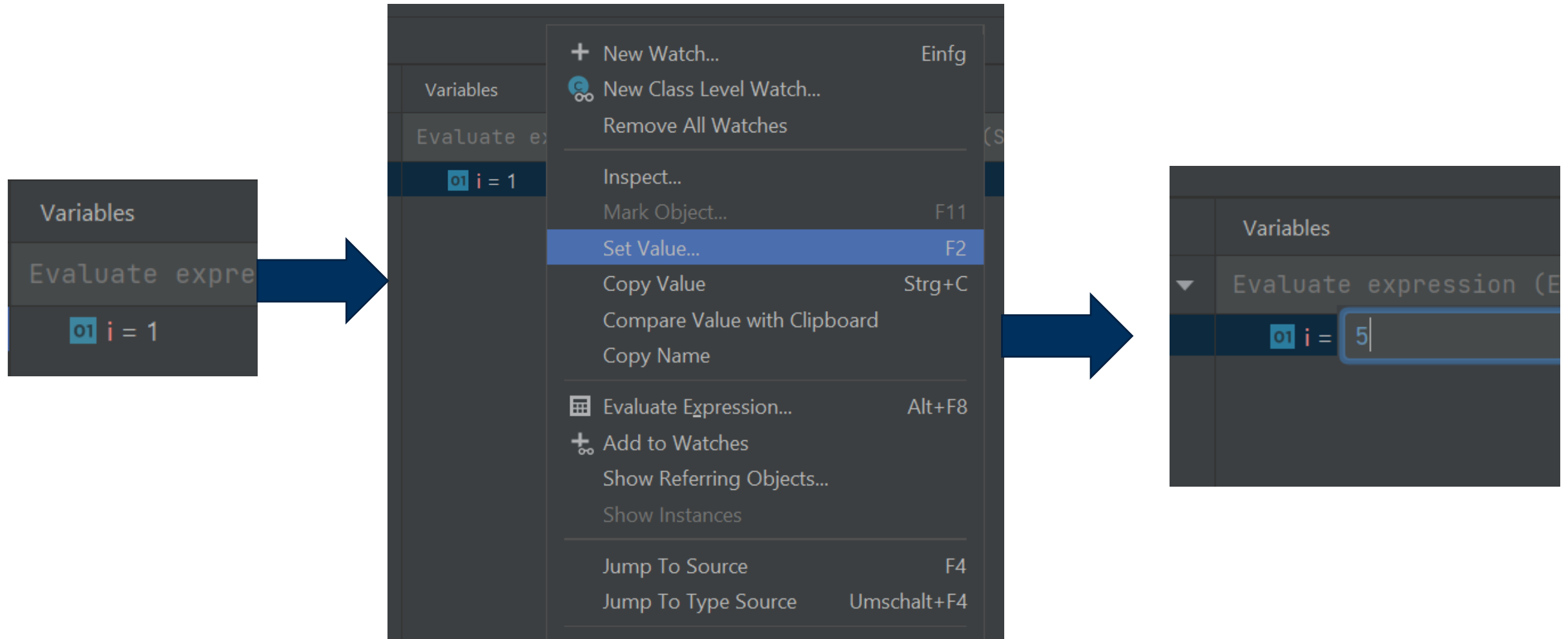
```
public static void level2() {  
    System.out.println("Start Level 2!\nViel Erfolg!");  
    final Level2Objekt zeus = new Level2Objekt( name: "Zeus", wert: 42, preis: 3.14); zeus: Lev  
    final Level2Objekt jupiter = new Level2Objekt( name: "Jupiter", wert: 42, preis: 3.14); jup  
    if (zeus.equals(jupiter)) { zeus: Level2Objekt@536 jupiter: Level2Objekt@538  
        // ...  
    }  
}
```

Variables

Evaluate expression (Eingabe) or add watch

- ▼ **zeus** = {Level2Objekt@536}
 - > **f** name = "Zeus"
 - f** wert = 42
 - f** preis = 3.14
- ▼ **jupiter** = {Level2Objekt@538}
 - > **f** name = "Jupiter"
 - f** wert = 42
 - f** preis = 3.14

Variablen-Fenster – Verändern von Werten



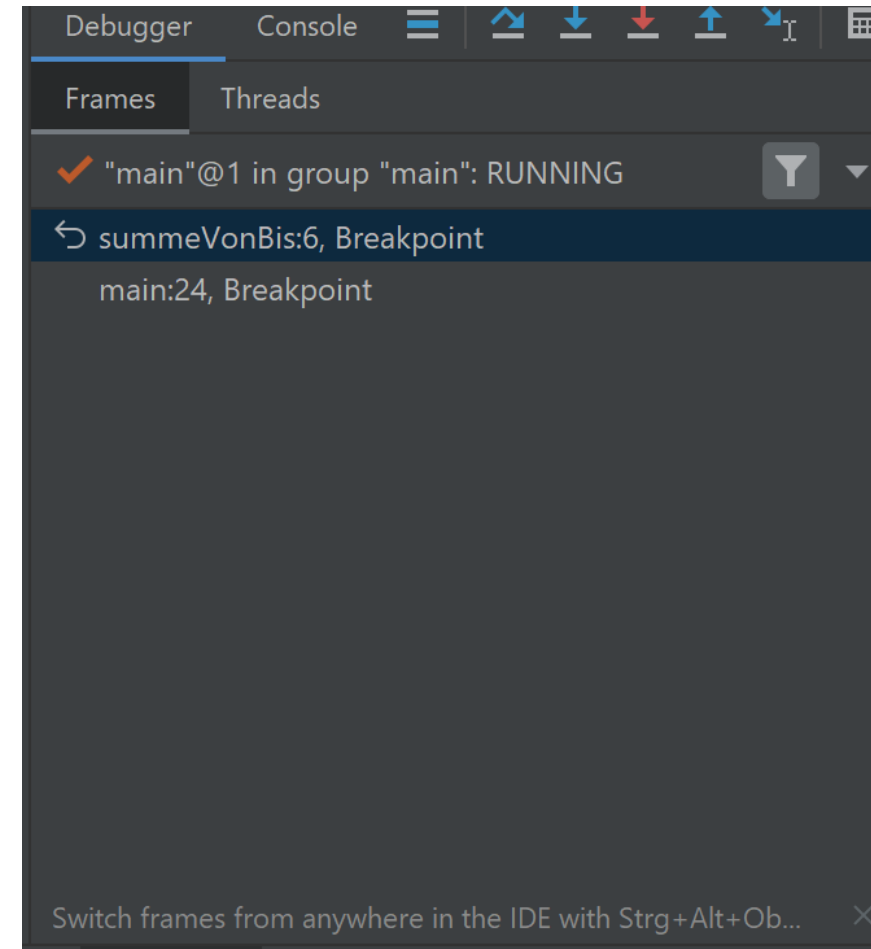
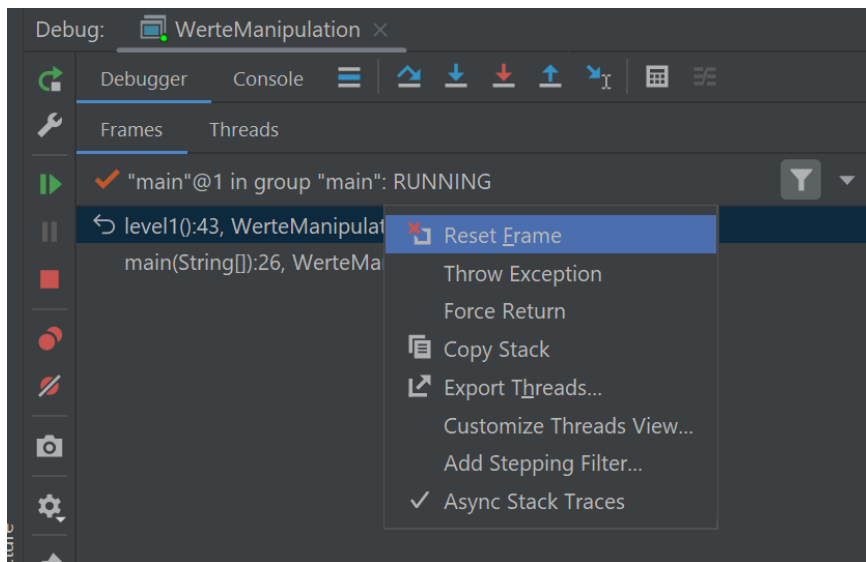
Frame Stack-Fenster

In Java-VM ist jede Methode in einem Frame – Siehe [3] 2.6

Dazu gehören die entsprechenden Variablen

Frames können durchsucht werden

Ein Frame kann auch verworfen werden



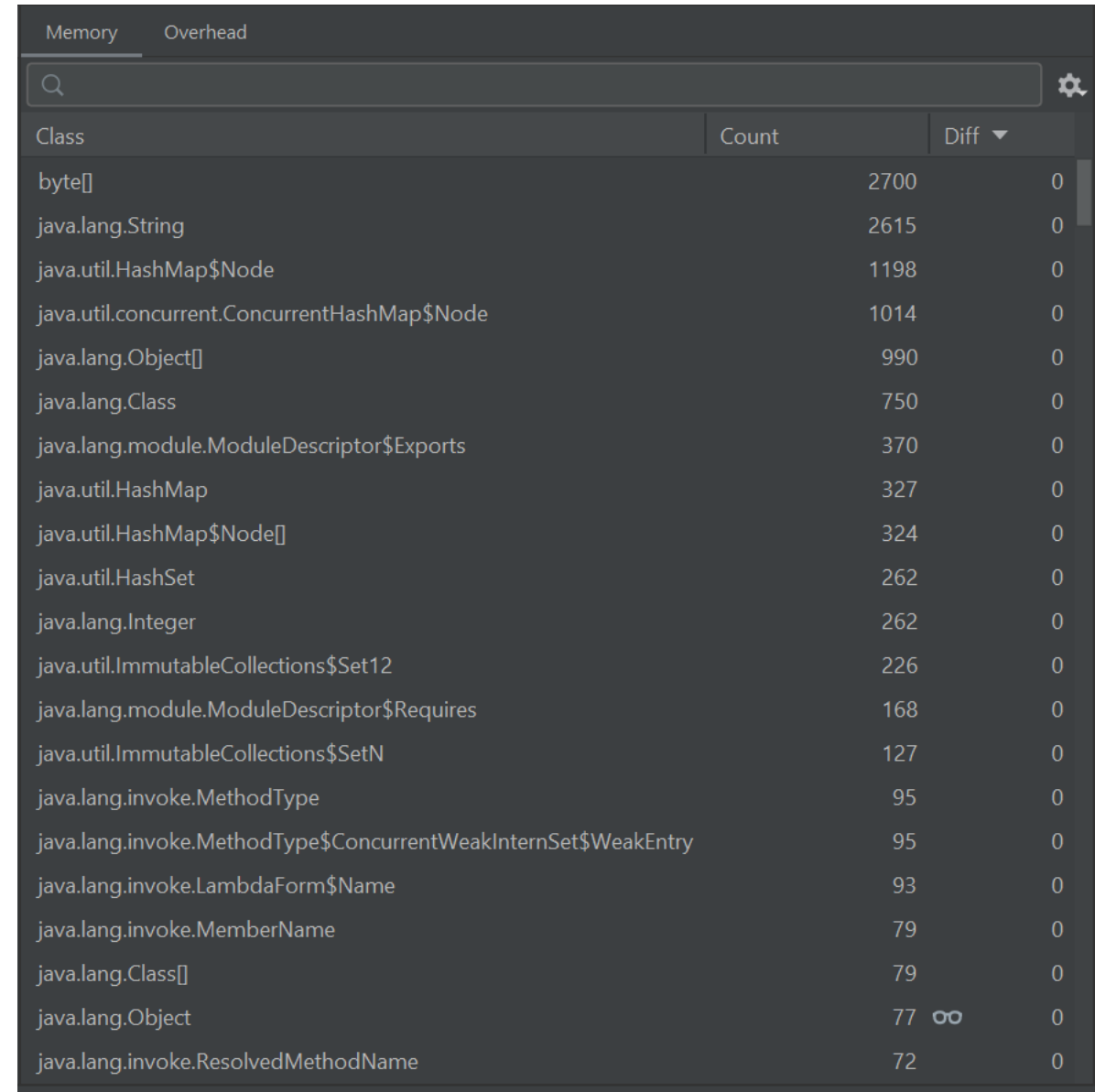
Extra Fenster - Memory

Zeigt *alle* Objekte im Speicher!

Sind nach Klasse sortiert.

Zeigt exakte Anzahl der Objekte

Diff = Änderung seit letztem Update



The screenshot shows the 'Memory' tab of an IDE's memory profiler. It displays a table of Java classes and their object counts. The table has three columns: 'Class', 'Count', and 'Diff'. The classes are sorted by count in descending order. The 'Diff' column shows the change in object count since the last update, with most entries being 0. The 'java.lang.Object' entry has a small icon next to its count.

| Class | Count | Diff |
|---|-------|------|
| byte[] | 2700 | 0 |
| java.lang.String | 2615 | 0 |
| java.util.HashMap\$Node | 1198 | 0 |
| java.util.concurrent.ConcurrentHashMap\$Node | 1014 | 0 |
| java.lang.Object[] | 990 | 0 |
| java.lang.Class | 750 | 0 |
| java.lang.module.ModuleDescriptor\$Exports | 370 | 0 |
| java.util.HashMap | 327 | 0 |
| java.util.HashMap\$Node[] | 324 | 0 |
| java.util.HashSet | 262 | 0 |
| java.lang.Integer | 262 | 0 |
| java.util.ImmutableCollections\$Set12 | 226 | 0 |
| java.lang.module.ModuleDescriptor\$Requires | 168 | 0 |
| java.util.ImmutableCollections\$SetN | 127 | 0 |
| java.lang.invoke.MethodType | 95 | 0 |
| java.lang.invoke.MethodType\$ConcurrentWeakInternSet\$WeakEntry | 95 | 0 |
| java.lang.invoke.LambdaForm\$Name | 93 | 0 |
| java.lang.invoke.MemberName | 79 | 0 |
| java.lang.Class[] | 79 | 0 |
| java.lang.Object | 77 | 0 |
| java.lang.invoke.ResolvedMethodName | 72 | 0 |

Weitere Hilfen

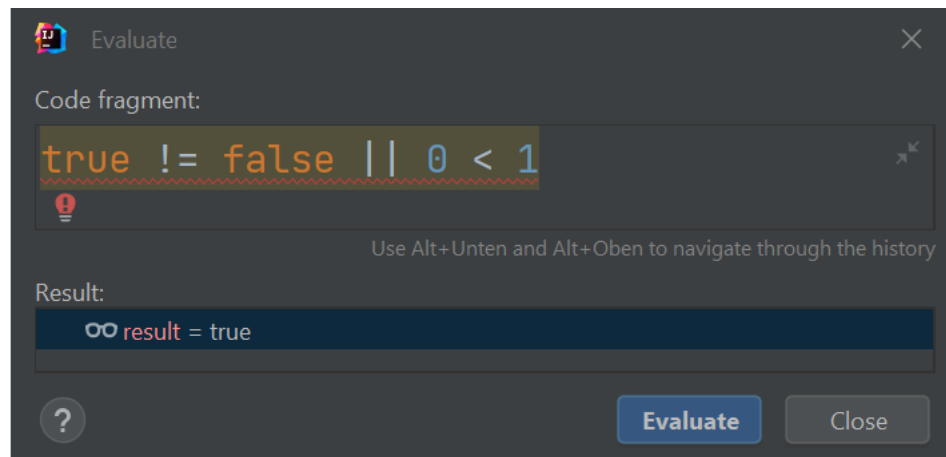
Expression Evaluator

(Alt + F8)

Kann jeden Ausdruck auswerten

Achtung kann Schaden anrichten!

```
if (a != 0
    && 5 % 2 == 1
    || true ^ false
    && (true != false || 0 < 1)) {
```

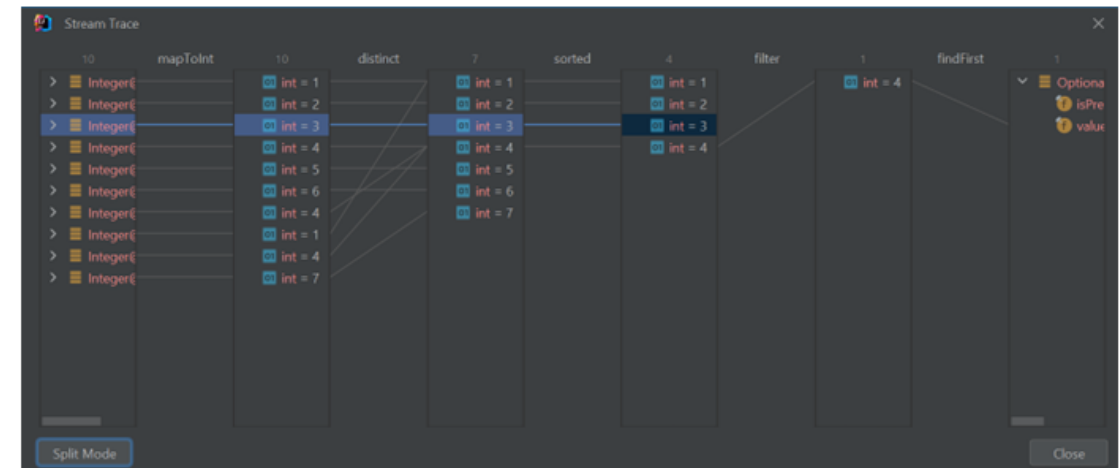


Chain Stream

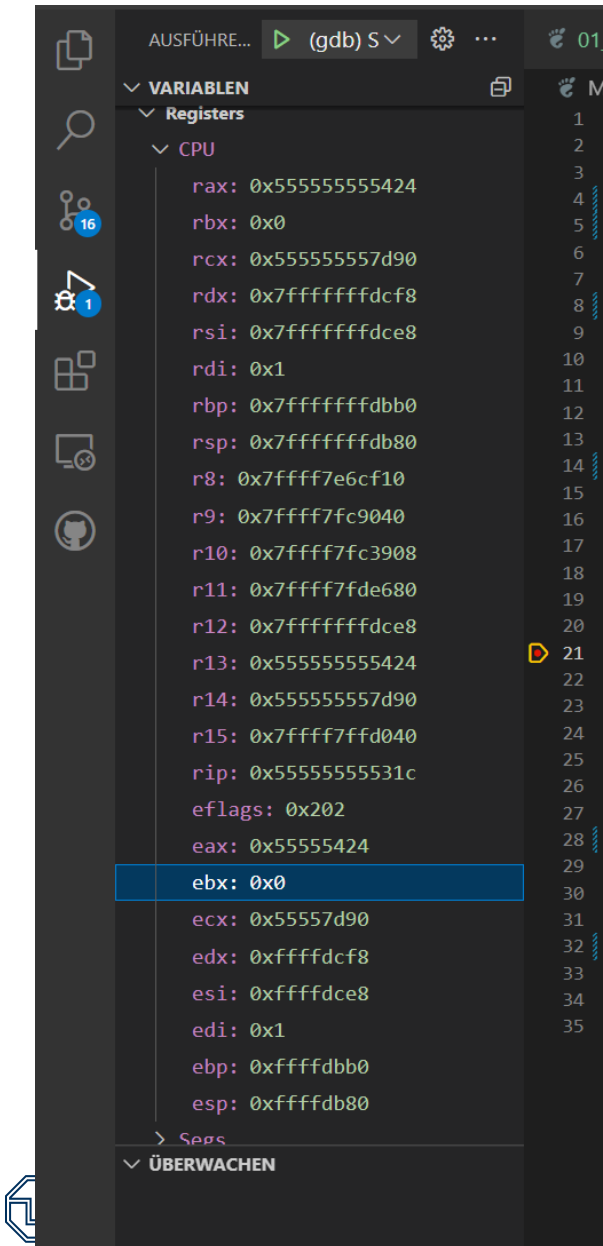
Benötigt ausführende Aktion

Nur in IntelliJ vorhanden

```
var test : Stream<Integer> = Stream.of(...values: 1, 2, 3, 4, 5, 6, 4, 1, 4, 7);
test.mapToInt(it -> it)
    .distinct()
    .sorted()
    .filter(it -> it > 3)
    .findFirst();
```



Andere Sprachen

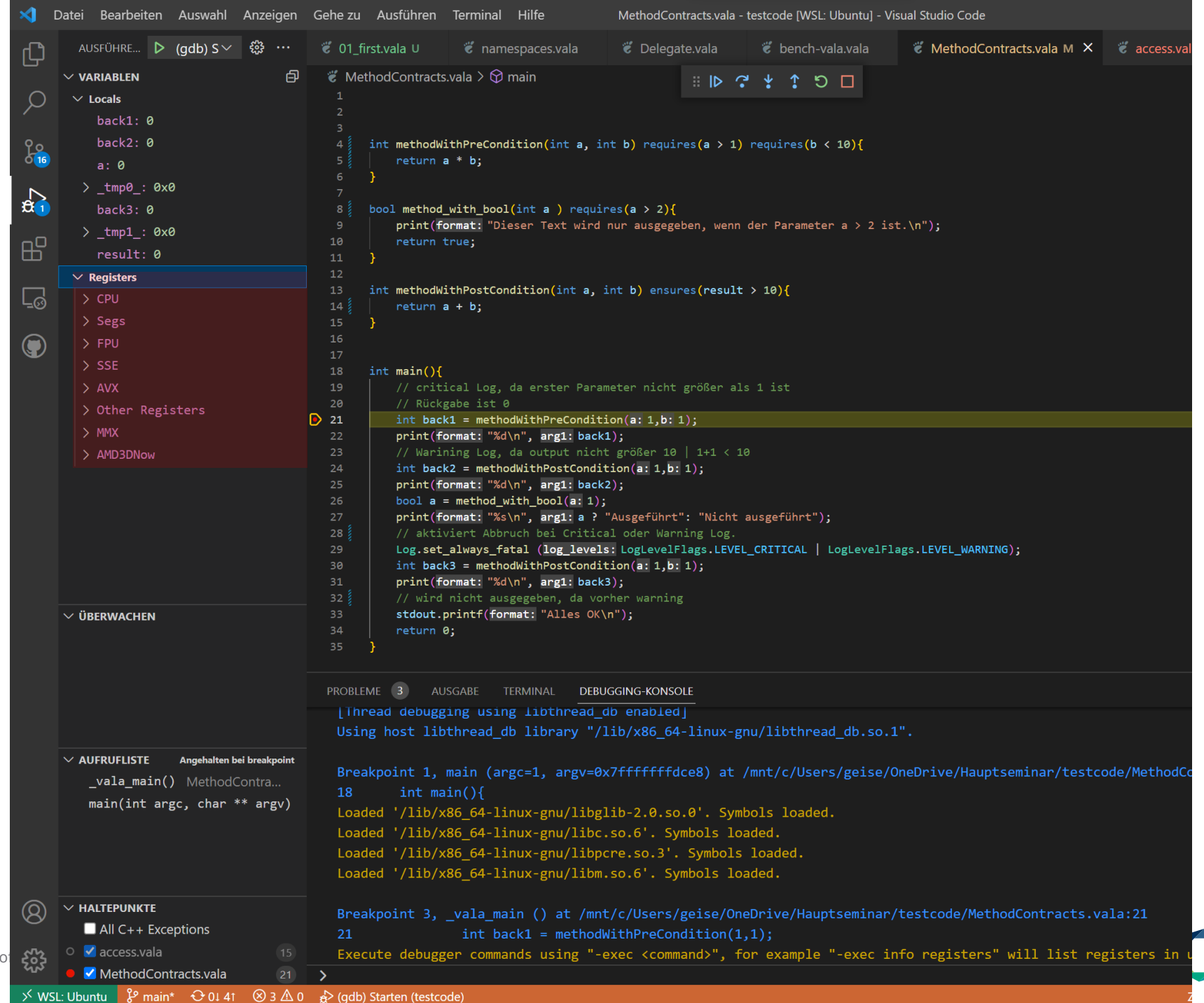


Visual Studio Code interface showing the GDB debugger. The 'Registers' pane is expanded, displaying CPU registers and their values. The 'Variables' pane is also visible, showing the current state of variables in the program.

VARIABLEN

- Registers**
 - CPU**
 - rax: 0x55555555424
 - rbx: 0x0
 - rcx: 0x555555557d90
 - rdx: 0x7fffffffdcf8
 - rsi: 0x7fffffffcdce8
 - rdi: 0x1
 - rbp: 0x7fffffffdbb0
 - rsp: 0x7fffffffdb80
 - r8: 0x7ffff7e6cf10
 - r9: 0x7ffff7fc9040
 - r10: 0x7ffff7fc3908
 - r11: 0x7ffff7fde680
 - r12: 0x7fffffffcdce8
 - r13: 0x55555555424
 - r14: 0x555555557d90
 - r15: 0x7ffff7ffd040
 - rip: 0x5555555531c
 - eflags: 0x202
 - eax: 0x555555424
 - ebx: 0x0**
 - ecx: 0x55557d90
 - edx: 0xffffdcf8
 - esi: 0xffffcdce8
 - edi: 0x1
 - ebp: 0xffffdbb0
 - esp: 0xffffdb80

ÜBERWACHEN



Visual Studio Code interface showing the GDB debugger. The 'Registers' pane is expanded, displaying CPU registers and their values. The 'Variables' pane is also visible, showing the current state of variables in the program.

VARIABLEN

- Locals**
 - back1: 0
 - back2: 0
 - a: 0
 - > _tmp0_: 0x0
 - back3: 0
 - > _tmp1_: 0x0
 - result: 0
- Registers**
 - CPU
 - Segs
 - FPU
 - SSE
 - AVX
 - Other Registers
 - MMX
 - AMD3DNow

ÜBERWACHEN

AUFRUFLLISTE Angehalten bei breakpoint

- _vala_main() MethodContra...
- main(int argc, char ** argv)

HALTEPUNKTE

- ☐ All C++ Exceptions
- ☒ access.vala
- ☒ MethodContracts.vala

MethodContracts.vala

```
1
2
3
4 int methodWithPreCondition(int a, int b) requires(a > 1) requires(b < 10){
5     return a * b;
6 }
7
8 bool method_with_bool(int a ) requires(a > 2){
9     print(format: "Dieser Text wird nur ausgegeben, wenn der Parameter a > 2 ist.\n");
10    return true;
11 }
12
13 int methodWithPostCondition(int a, int b) ensures(result > 10){
14    return a + b;
15 }
16
17
18 int main(){
19     // critical Log, da erster Parameter nicht größer als 1 ist
20     // Rückgabe ist 0
21     int back1 = methodWithPreCondition(a: 1,b: 1);
22     print(format: "%d\n", arg1: back1);
23     // Warning Log, da output nicht größer 10 | 1+1 < 10
24     int back2 = methodWithPostCondition(a: 1,b: 1);
25     print(format: "%d\n", arg1: back2);
26     bool a = method_with_bool(a: 1);
27     print(format: "%s\n", arg1: a ? "Ausgeführt": "Nicht ausgeführt");
28     // aktiviert Abbruch bei Critical oder Warning Log.
29     Log.set_always_fatal (log_levels: LogLevelFlags.LEVEL_CRITICAL | LogLevelFlags.LEVEL_WARNING);
30     int back3 = methodWithPostCondition(a: 1,b: 1);
31     print(format: "%d\n", arg1: back3);
32     // wird nicht ausgegeben, da vorher warning
33     stdout.printf(format: "Alles OK\n");
34     return 0;
35 }
```

PROBLEME 3 **AUSGABE** **TERMINAL** **DEBUGGING-KONSOLE**

[hread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main (argc=1, argv=0x7fffffffcdce8) at /mnt/c/Users/geise/OneDrive/Hauptseminar/testcode/MethodContracts.vala:18

```
18 int main(){
Loaded '/lib/x86_64-linux-gnu/libglib-2.0.so.0'. Symbols loaded.
Loaded '/lib/x86_64-linux-gnu/libc.so.6'. Symbols loaded.
Loaded '/lib/x86_64-linux-gnu/libpcrcr.so.3'. Symbols loaded.
Loaded '/lib/x86_64-linux-gnu/libm.so.6'. Symbols loaded.
```

Breakpoint 3, _vala_main () at /mnt/c/Users/geise/OneDrive/Hauptseminar/testcode/MethodContracts.vala:21

```
21 int back1 = methodWithPreCondition(1,1);
Execute debugger commands using "-exec <command>", for example "-exec info registers" will list registers in u
```

Andere Sprachen

Datei Bearbeiten Ansicht Git Projekt Erstellen Debuggen Test Analysieren Extras

Prozess: [8272] N-Body-Cuda.exe Ereignisse des Lebenszyklus Thread: [1] CUDA Thread

Start CUDA Debugging (Next-Gen)

Projektmappen-Explorer

compilescript CMakeLists.txt main.cu

C++ IntelliSense-Informationen sind möglicherweise veraltet, ger

Projektmappen-Explorer durch

Projektmappe "N-Body-Cuda" (

N-Body-Cuda

Externe Abhängigkeiten

x64

nvsettings

CMakeLists.txt

main.cu

N-Body-Cuda.sln.nvsetting

shared_header.h

```
109
110 // iterates through the particles-ar
111 // and for each (of three) particle
112 //
113 global void aos_move(particle* p
114 int id = LINEAR_ID;
115 if (id < PROBLEMSIZE) {
116     for (int j = 0; j < 3; ++j)
117         particles[id].pos[j] +=
118     }
119 }
120
121
122
123
124 global void aos_randNormal(struc
125 int tid = LINEAR_ID;
126 curandState state;
127 curand_init(1337, tid, 0, &state
128
129 // write random values to partic
130 struct particle p = particle[tid
131
132 for (int i = 0; i < 3; ++i) {
133     p.pos[i] = curand_normal(&st
134     p.vel[i] = curand_normal(&st
135 }
136 p.mass = curand_normal(&state);
137
138 particle[tid] = p;
```

Überwachen 1

Suchen (Strg+E)

Name Wert

particles Der Ausdruck

particles:28 Der Ausdruck

Element zur Überwachung hinzufügen

array_of_structs.cu

Lanes

Enter filter

| Lane | Thread Index | Status | PC | Exception |
|------|--------------|------------|----------|-----------|
| 0 | (0, 0, 0) | Breakpoint | 0040b8b0 | None |
| 1 | (1, 0, 0) | Active | 0040b8b0 | None |
| 2 | (2, 0, 0) | Active | 0040b8b0 | None |
| 3 | (3, 0, 0) | Active | 0040b8b0 | None |
| 4 | (4, 0, 0) | Active | 0040b8b0 | None |
| 5 | (5, 0, 0) | Active | 0040b8b0 | None |
| 6 | (6, 0, 0) | Active | 0040b8b0 | None |
| 7 | (7, 0, 0) | Active | 0040b8b0 | None |
| 8 | (8, 0, 0) | Active | 0040b8b0 | None |
| 9 | (9, 0, 0) | Active | 0040b8b0 | None |
| 10 | (10, 0, 0) | Active | 0040b8b0 | None |
| 11 | (11, 0, 0) | Active | 0040b8b0 | None |
| 12 | (12, 0, 0) | Active | 0040b8b0 | None |
| 13 | (13, 0, 0) | Active | 0040b8b0 | None |
| 14 | (14, 0, 0) | Active | 0040b8b0 | None |

Lanes Diagnosetools

GPU Registers

Local

R0 = bdf8b3ec R1 = 00ffffe0 R2 = 00000174 R3 = 00000174 R4 = bdf8b3ec R5 = 00000174
R6 = 00000174 R7 = 00000174 R8 = 9bffffd4 R9 = 00000174 R10 = 00000174 R11 = 00000174
R12 = 00000000 R13 = 00000000 R14 = 00000000 R15 = 00000000 R16 = 9bffffe0 R17 = 0000000b
R18 = 04e00000 R19 = 00000000 R20 = 3ea94f62 R21 = 404e4abf R22 = 3f8e6c3b R23 = 83545ca7
R24 = 54a7b132 R25 = 00000001 R26 = 9bffffd5 R27 = 00000174 R28 = 3d40c0c1 R29 = 3d40c0c1
R30 = 3d40c0c1 R31 = 00000000 R32 = 00000000 R33 = 00000539 R34 = 9bffffd9 R35 = 00000174
R36 = 3d40c0c1 R37 = 3d40c0c1 R38 = 3d40c0c1 R39 = 3d40c0c1 R40 = 3d40c0c1 R41 = 3d40c0c1
R42 = 3d40c0c1 R43 = 3d40c0c1 R44 = 3d40c0c1 R45 = 00000000 R46 = 00000000 R47 = 3d40c0c1
R48 = 3d40c0c1 R49 = 3d40c0c1 R50 = 3d40c0c1 R51 = 3d40c0c1 R52 = 00000000 R53 = 00000000
R54 = 00000000 R55 = 00000000

Predicate

P0 = 1 P1 = 1 P2 = 1 P3 = 1 P4 = 0 P5 = 0 P6 = 0 PT = 1

GPU Registers

Resources

Zeile: 133 Zeichen: 1 GEMISCHT CRLF

Warp Info

Enter filter

| Context | SM Version | Grid ID | Shader Info | Threads | PC | Active Mask |
|------------|------------|----------|---------------------------------------|----------|------|-------------|
| 17499254d0 | 00060001 | 00000001 | CTA: (0, 0, 0), Thread: (0, 0, 0) | 0040b8b0 | FFFI | |
| 17499254d0 | 00060001 | 00000001 | CTA: (0, 0, 0), Thread: (32, 0, 0) | 00405648 | FFFI | |
| 17499254d0 | 00060001 | 00000001 | CTA: (0, 0, 0), Thread: (64, 0, 0) | 00405648 | FFFI | |
| 17499254d0 | 00060001 | 00000001 | CTA: (0, 0, 0), Thread: (96, 0, 0) | 00405648 | FFFI | |
| 17499254d0 | 00060001 | 00000001 | CTA: (0, 0, 0), Thread: (128, 0, 0) | 00405648 | FFFI | |
| 17499254d0 | 00060001 | 00000001 | CTA: (0, 0, 0), Thread: (160, 0, 0) | 00405648 | FFFI | |
| 17499254d0 | 00060001 | 00000001 | CTA: (0, 0, 0), Thread: (192, 0, 0) | 00405648 | FFFI | |
| 17499254d0 | 00060001 | 00000001 | CTA: (0, 0, 0), Thread: (224, 0, 0) | 00405648 | FFFI | |
| 17499254d0 | 00060001 | 00000001 | CTA: (0, 0, 0), Thread: (256, 0, 0) | 00405648 | FFFI | |
| 17499254d0 | 00060001 | 00000001 | CTA: (0, 0, 0), Thread: (288, 0, 0) | 00405648 | FFFI | |

Warp Watch

Enter filter

| Value | <Add Watch> |
|-------|-------------|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |

Warp Watch

Enter filter

| Status | PC | Exception |
|--------|----------|-----------|
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |

Warp Watch

Enter filter

| Threads | PC | Active Mask |
|----------------------|----------|-------------|
| Thread: (0, 0, 0) | 00408770 | FFFI |
| Thread: (32, 0, 0) | 00405648 | FFFI |
| Thread: (64, 0, 0) | 00405648 | FFFI |
| Thread: (96, 0, 0) | 00405648 | FFFI |
| Thread: (128, 0, 0) | 00405648 | FFFI |
| Thread: (160, 0, 0) | 00405648 | FFFI |
| Thread: (192, 0, 0) | 00405648 | FFFI |
| Thread: (224, 0, 0) | 00405648 | FFFI |
| Thread: (256, 0, 0) | 00405648 | FFFI |

Anmelden

Live Share

Warp Watch

Enter filter

| Value | <Add Watch> |
|-------|-------------|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |

Warp Watch

Enter filter

| Status | PC | Exception |
|--------|----------|-----------|
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |
| 0 | 00408770 | None |

Warp Watch

Enter filter

| Threads | PC | Active Mask |
|----------------------|----------|-------------|
| Thread: (0, 0, 0) | 00408770 | FFFI |
| Thread: (32, 0, 0) | 00405648 | FFFI |
| Thread: (64, 0, 0) | 00405648 | FFFI |
| Thread: (96, 0, 0) | 00405648 | FFFI |
| Thread: (128, 0, 0) | 00405648 | FFFI |
| Thread: (160, 0, 0) | 00405648 | FFFI |
| Thread: (192, 0, 0) | 00405648 | FFFI |
| Thread: (224, 0, 0) | 00405648 | FFFI |
| Thread: (256, 0, 0) | 00405648 | FFFI |

Tests

Prüft zur Laufzeit, ob ein bestimmter **Fehler** auftrat

Geben die Abweichung vom **Sollzustand** an.

Geben nicht die **Ursache** an

Jeder Test sollte eine **Fehler-Message** besitzen



Einordnung von Tests:

Nach „Stufen“

- Unit-Test
- Integration-Test
- System-Test
- Abnahme-Test

(Siehe V-Modell)

Nach „Akteur“

- Manuel
- Automatisch

Nach Art

- Funktional
- Nicht funktional
- Änderungsbezogen
- White-Box
(nach [5])

Nach „Wissen“

- White-Box
- Gray-Box
- Black-Box

Tests können mehr als public testen

Gängige Praxis:

Klassen in: src.project.section.MeineKlasse.java

Tests in: test.project.section.MeineKlasseTest.java

Beide Klassen sind im package project.section

Dadurch hat MeineKlasseTest.java auf alle *public*, *package* und *protected* Methoden/Attribute Zugriff

```
package de.oliver.structure;
...
public abstract class Arbeitsplatz<T extends Person> implements Verschmutzbar {

    protected T nutzer;
    protected double verschmutzung;
```

```
package de.oliver.structure;
...
import static org.junit.jupiter.api.Assertions.*;

class ArbeitsplatzTest {

    @Test
    void isDreckigTrue(){
        assertFalse(arbeitsplatz.isDreckig(),"Am Anfang darf es nicht dreckig sein");
        var t =arbeitsplatz.verschmutzung;
        arbeitsplatz.verschmutzen();
```

Tests müssen nicht private testen

Private steuert das innere Verhalten des Objektes. Es ist für alle anderen unwichtig.

Private Methoden sollen helfen, das äußere Verhalten zu erreichen.

Private Methoden werden durch nicht private Methoden aufgerufen

Wenn *private* explizit getestet werden muss, dann ist es oft schlechtes Design!

Reflection erlaubt das Testen der privaten Methoden, aber es sollte sparsam eingesetzt werden.

JUNIT 5.8.0 [4] – Neue Features

`assertX(Sollwert, Istwert, „Nachricht bei Fehler“)`

`assertAll` – alle enthaltenen `assertX` werden ausgeführt und zusammen ausgegeben.

Liste der Annotationen: <https://junit.org/junit5/docs/current/user-guide/#writing-tests-annotations>

`@ParameterizedTest` – „Template“ für viele Testfälle

`@DisplayName` – Name im Testergebnis

Tests in einer IDE

IDEs haben integrierte Test-Umgebungen

Liefern Übersicht der Ergebnisse

Können Tests automatisch starten

| | |
|---|---------------|
| ! <default package> | 47 sec 74 ms |
| > ✓ ArbeitsplatzTest | 88 ms |
| > ✓ AngestellterTest | 52 ms |
| > ✗ BreakpointTest | 24 ms |
| ▼ ! RegalTest | 46 sec 695 ms |
| ✗ isVollFalseBeimErzeugen() | 1 ms |
| ! alleBuecherOkay() | 46 sec 628 ms |
| ✗ leereVollesRegal() | 29 ms |
| ! enthaeltBuchOkay() | 2 ms |
| ✓ nullBuecherAmAnfang() | 31 ms |
| ✓ iteratorTest() | 1 ms |
| ✓ toStringTest() | |
| ✓ enthaeltNullTest() | 1 ms |
| ✓ isVollTrue() | 1 ms |
| ✗ isVollFalseBeimEntnehmenEinesBuches | 1 ms |
| > ! TestErgebnisse | 6 ms |
| > ✓ ISBNTest | 148 ms |
| org.opentest4j.AssertionFailedError: Die Ausleihe eines Buches darf nicht gelingen, wenn es ber | 61 ms |

org.opentest4j.AssertionFailedError: Die Ausleihe eines Buches darf nicht gelingen, wenn es ber

<3 internal lines>

at de.oliver.core.BuchTest.ausleihenFehlslag(BuchTest.java:27) <31 internal lines>

at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>

at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <27 internal lines>

Arten von Testergebnissen

Test ist erfolgreich.
Keine Fehler.

Test ist fehlgeschlagen.
Ergebnis stimmt nicht.

Test ist fehlgeschlagen.
Explizit durch den Entwickler!

Der Test wurde unerwartet unterbrochen.
Eine Exception trat auf.

Test wird als abgebrochen markiert

Test wurde ignoriert.

```
import static org.junit.jupiter.api.Assertions.*;
import static org.junit.jupiter.api.Assumptions.assumeFalse;

public class TestErgebnisse {

    @Test
    public void erfolgreicherTest(){
        assertTrue( condition: true, message: "Der Test erfordert true");
    }

    @Test
    public void fehlerhafterTest(){
        assertTrue( condition: false, message: "Der Test erfordert true");
    }

    @Test
    public void fehlgeschlagenerTest(){
        fail("Dieser Test schlägt immer Fehl!");
    }

    @Test
    public void TestMitUnerwarteterException(){
        throw new IllegalArgumentException("Unerwartete Exception");
    }

    @Test
    public void abgebrochenerTest(){
        assumeFalse( assumption: true, message: "Die Annahme ist nicht Falsch!");
    }

    @Test
    @Disabled
    public void übersprungenerTest(){
        // Dieser Test wird nicht ausgeführt
    }
}
```

Arten von Testergebnissen

Run: TestErgebnisse x

Tests failed: 3, passed: 1, ignored: 2

| Testergebnis | Dauer | Details |
|--------------------------------|-------|--|
| fehlerhafterTest() | 33 ms | org.opentest4j.AssertionFail Expected :true Actual :false Click to see difference |
| erfolgreicherTest() | 1 ms | |
| abgebrochenerTest() | 6 ms | |
| übersprungenerTest() | | |
| fehlgeschlagenerTest() | 1 ms | |
| TestMitUnerwarteterException() | 2 ms | <3 internal lines> at TestErgebnisse.fehlerhafterTest() at java.base/java.util.A at java.base/java.util.A |

```
import static org.junit.jupiter.api.Assertions.*;
import static org.junit.jupiter.api.Assumptions.assumeFalse;

public class TestErgebnisse {

    @Test
    public void erfolgreicherTest(){
        assertTrue( condition: true, message: "Der Test erfordert true");
    }

    @Test
    public void fehlerhafterTest(){
        assertTrue( condition: false, message: "Der Test erfordert true");
    }

    @Test
    public void fehlgeschlagenerTest(){
        fail("Dieser Test schlägt immer Fehl!");
    }

    @Test
    public void TestMitUnerwarteterException(){
        throw new IllegalArgumentException("Unerwartete Exception");
    }

    @Test
    public void abgebrochenerTest(){
        assumeFalse( assumption: true, message: "Die Annahme ist nicht Falsch!");
    }

    @Test
    @Disabled
    public void übersprungenerTest(){
        // Dieser Test wird nicht ausgeführt
    }
}
```

Die Bibliothek - Anforderungen

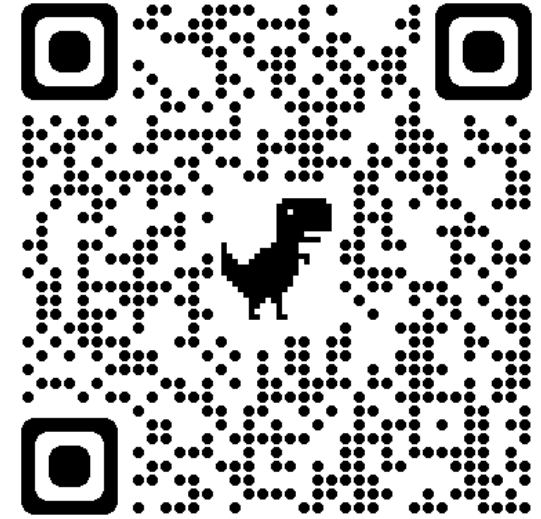
Maven

Java 19

Eine IDE wie Eclipse, IntelliJ, VS Code usw.

Download des Repos von GitHub

Entweder lokaler Import oder direkt vom Server in die IDE



Die Bibliothek - Situation

Aus den ersten Übungen ist die Bibliothek bereits bekannt.

Diese ist inzwischen zu einer großen Anwendung angewachsen.

Es gibt Personal- und Besucher-Verwaltung sowie Leseräume.

Zwei SHKs haben diese Anwendung gebaut.

Der eine hat die Tests, der andere den Quellcode geschrieben.

Leider sind im Quellcode Fehler. Dies wurde durch die Tests herausgefunden.

Die Bibliothek - Aufgabe

Finden der Bugs im Projekt anhand der Tests und durch anwenden von Debugging.

Hinweise:

- **Die Tests sind korrekt**
- **Alle Stellen, die Features ab Java 8 haben, sind korrekt**
- **Die neuen Features sollen Motivation sein, sich selbst mit neuen Features in Java zu beschäftigen**
- **Weitere Information finden Sie im GitHub-Repository in der Datei *readme.md***

2 Phasen:

- 1. Mit Tests die Fehler eingrenzen**
- 2. Durch Debugging die fehlerhaften Stellen suchen und beheben**

Elemente, die in Java 8+ neu sind

- **Var (Type Inference) – 10**
- **Optional – 8**
- **Stream – 8**
- **Lambda – 8**
- **Records – 14/16**
- **Pattern matching for switch – 17/20+**
- **Pattern matching for instanceof – 14/16**
- **Enhanced Switch – 12/14**
- **Switch-Expression – 12/14**
- **Textblocks – 13/15**

Die Bibliothek – Tests analysieren

Run: All in Debug-Tutorial-bibliothek x

Tests failed: 42, passed: 144, ignored: 2 of 188 tests – 2 sec 788 ms

| Test | Duration |
|----------------------------|--------------|
| <default package> | 2 sec 788 ms |
| AngestelltenComputerTest | 8 ms |
| AngestelltenVerwaltungTest | 13 ms |
| AngestellterTest | 23 ms |
| ArbeitsplatzTest | 116 ms |
| BestandsVerwaltungTest | 199 ms |
| BesucherComputerTest | 4 ms |
| BesucherTest | 10 ms |
| BibliothekTest | 7 ms |
| BuchTest | 42 ms |
| DozentTest | |
| ISBNTest | 121 ms |
| KundenregisterTest | 210 ms |
| LeseraumTest | 81 ms |
| RegalTest | 75 ms |
| StudierenderTest | 2 ms |
| WerkstattTest | 1 sec 877 ms |

Expected :false
Actual :true
[Click to see difference](#)

org.opentest4j.AssertionFailedError
at de.oliver.structure.RegalTest
at java.base/java.util.ArrayList
at java.base/java.util.ArrayList

| | | |
|---|--|--------|
| > | BibliothekTest | 7 ms |
| > | BuchTest | 42 ms |
| > | KundenregisterTest | 210 ms |
| > | strafeFuerDozent(int, double) | 138 ms |
| > | bezahlenOkay() | 27 ms |
| > | getAusgelieheneBuecherOkay() | 1 ms |
| > | getStrafe1TagVorAbgabe() | 1 ms |
| > | getStrafe7TageVorAbgabe() | 2 ms |
| > | getStrafeAmAbgabeTag() | 1 ms |
| > | getStrafeNach1Tag() | 10 ms |
| > | getStrafeNach2Jahren() | 2 ms |
| > | getStrafeNach7Tagen() | 1 ms |
| > | getStrafeNach8Tagen() | 2 ms |
| > | getStrafeNach14Tagen() | 1 ms |
| > | getStrafeNach15Tagen() | 1 ms |
| > | getStrafeNach39Wochen() | 2 ms |
| > | getStrafeNach40Wochen() | 2 ms |
| > | getStrafeNach41Wochen() | 1 ms |
| > | getStrafeNach43Tagen() | 1 ms |
| > | gibBuchZurueckFehlerNichtDemBesucherZugr | 1 ms |
| > | gibBuchZurueckOkay() | 1 ms |
| > | leiheBuchAusOkay() | 1 ms |

```
java.lang.NullPointerException Create breakpoint : Cannot invoke "java.util.List.add(Object)" because "this.ausgelieheneBuecher" is null
    at de.oliver.person.visitor.Kundenregister$BesucherStatus.registriereAusgeliehenesBuch(Kundenregister.java:155)
    at de.oliver.person.visitor.Kundenregister.leiheBuchAus(Kundenregister.java:62)
    at de.oliver.person.visitor.KundenregisterTest.getAusgelieheneBuecherOkay(KundenregisterTest.java:276) <31 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <27 internal lines>
```

Lösung

Im „Lösung“-Branch sind die Lösungen zu finden. Diese sind mit Todo´s markiert.

19 Bugs können gefunden werden

Mögliches Refactoring:

- Singleton
- Abstrakte Klassen einfügen
- State-Pattern
- Observer

Fürs Selbststudium

Findet schlechte Ideen und verbessert Sie

Nutzt SonarQube/SonarLint für Codequalität

Erweitert die Bibliothek

Nutzt TDD (Test-Driven-Development)

Baut für Bücher das Interface Zerstörbar

Baut bei ISBN die Prüfzifferkontrolle

Nutzt Logging

Baut Pattern ein

Multithreading von piNaehern()

Literatur

Java:

- [1] <https://docs.oracle.com/en/java/javase/18/docs/api/index.html> - JavaDoc 18
- [2] <https://docs.oracle.com/javase/specs/jls/se18/jls18.pdf> - Java Language Specification
- [3] <https://docs.oracle.com/javase/specs/jvms/se18/jvms18.pdf> - Java VM Specification
- [4] <https://junit.org/junit5/docs/current/user-guide/> - JUnit 5 User Guide

Testen:

- [5] <https://www.german-testing-board.info/lehrplaene/istqbr-certified-tester-schema/entwicklungstester/> - Lehrplan des ISTQB
- [6] <https://ieeexplore.ieee.org/document/5399061> - Klassifikation von Software Anomalien

Debugging:

- [7] <https://www.jetbrains.com/help/idea/debugging-code.html> - IntelliJ Debugger
- [8] https://www.eclipse.org/community/eclipse_newsletter/2017/june/article1.php - Eclipse Debugger Einführung
- [9] <https://help.eclipse.org/latest/nav/80> - Debugging in Eclipse

Weiteres Material

Bonus – TDD nutzen

Nach Robert C Martin gibt es drei Regeln des TDD:

1. „You are not allowed to write any production code unless it make failing unittest pass.“
-> Du Brauchst einen fehlschlagenden Test um Code zu schreiben!
2. „You are not allowed to write any more of a unit test than is sufficient to fail; and compilation failures are failuers“
-> Du darfst nicht mehr als einen fehlschlagenden Test schreiben!
3. „You are not allowed to write any more production code than is sufficent to pass the one failing unit test.“
-> Schreib nur Code, der den einen Test erfüllt!

Siehe:

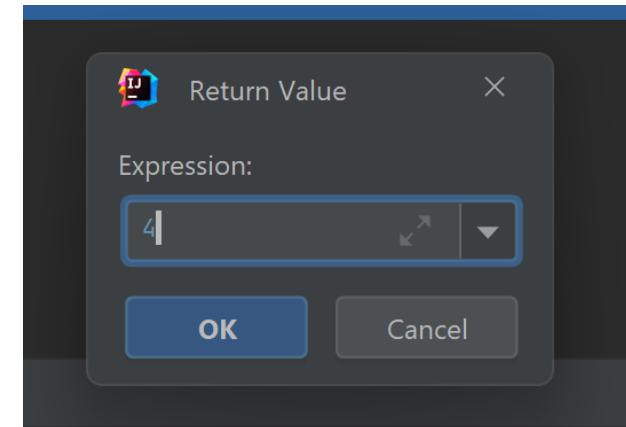
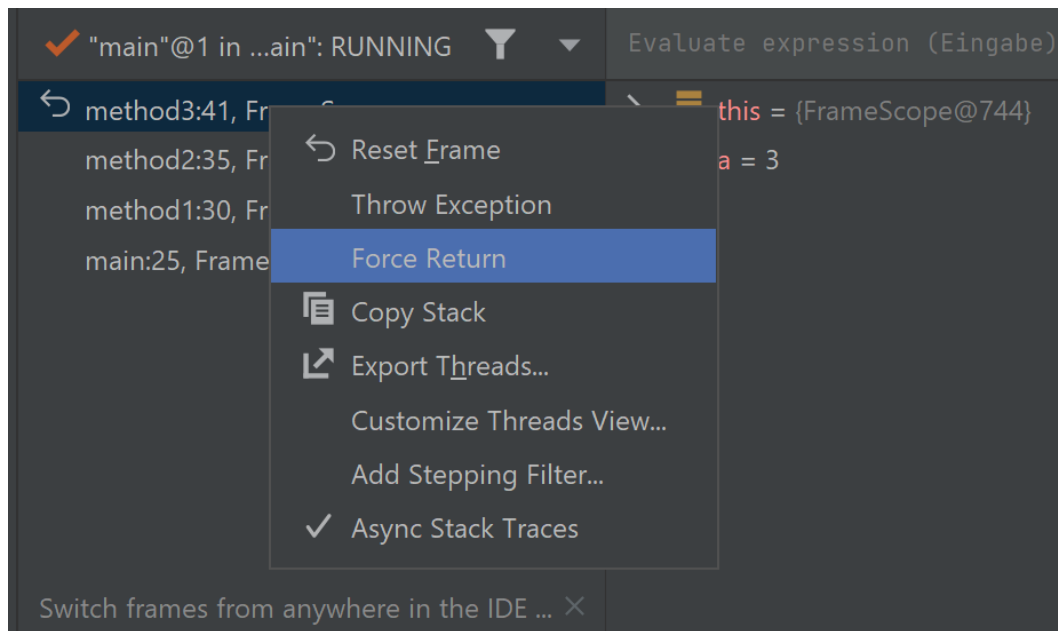
<https://www.youtube.com/watch?v=qkblc5WRn-U>

<https://youtu.be/58jGpV2Cg50?t=1300>

Force Return

Innerhalb einer Methode kann jederzeit abgebrochen werden

Angabe eines Rückgabewertes benötigt, wenn Rückgabotyp nicht *void* ist



Jump to line intellij

Ein Plugin in der IntelliJ IDE

Erlaub das Überspringen von Befehlen

Durch Wahl einer Zeile wird jeder Befehl von der aktuellen Zeile an übersprungen

jShell – Konsole zum schnellen testen

Neu seit Java 9

Erlaubt Entwicklung als laufender Anwendung

Klassen werden in der Kommandozeile definiert

Ähnlich wie Python (das Kommando)

Gut zum schnellen Testen