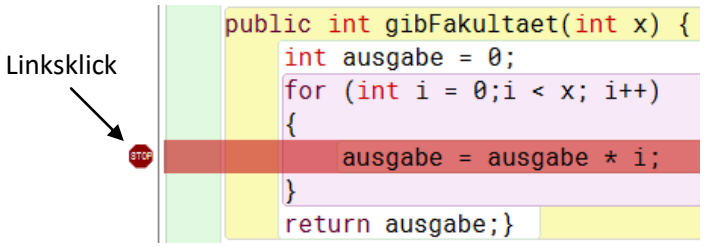
Debugging - Fehler finden

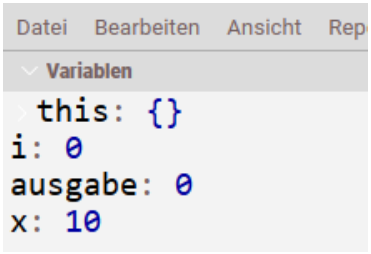
Möglichkeit 1: Breakpoint

Beispiel: In folgendem JAVA-Code soll die Fakultät einer Zahl x berechnet werden, doch die Ergebnisse sind immer falsch. `gibFakultaet(5)` liefert leider `0`.

```
public int gibFakultaet(int x) {
int ausgabe = 0;
for (int i = 0;i < x; i++)
{
   ausgabe = ausgabe * i;
}
return ausgabe;}
```



Setzt man eine Unterbrechung (Breakpoint) an der fraglichen Stelle des JAVA-Code, kann beim Ausführen der Methode die aktuelle Belegung der Variablen und Objekte an dieser Stelle betrachtet werden.



Mit Hilfe von  kann man das Programm bis zum nächsten Breakpoint laufen lassen, Veränderungen beobachten und evtl. Fehler finden. Auch durch Regelung der Geschwindigkeit  oder Einzelschritt  kann man so Fehler finden.

Im Beispiel: Man sieht, dass sich `ausgabe` nicht erhöht. (siehe JAVA-Code online-IDE)

Möglichkeit 2: System.out.println bzw. println (nur in der online-IDE)

Setzt man an der fraglichen Stelle des JAVA-Code eine Systemausgab, welche die aktuellen Variablen liefert, kann nach dem Ausführen evtl. Fehler erkennen und beheben.

```
public int gibFakultaet(int x) {
int ausgabe = 0;
for (int i = 0;i < x; i++)
{
   ausgabe = ausgabe * i;
   println("i: "+i+" Ausgabe: "+ausgabe+" x:"+x);
}
return ausgabe;}
```

Konsolenausgabe des Beispiels:

```
r.gibFakultaet(5)
i: 0 Ausgabe: 0 x:5
i: 1 Ausgabe: 0 x:5
i: 2 Ausgabe: 0 x:5
i: 3 Ausgabe: 0 x:5
i: 4 Ausgabe: 0 x:5
    returned int 0
```

Im Beispiel: Auch hier erhöht sich `ausgabe` nicht.