

# † bung: ZŠhl-Threads

## Schritt 1

Erstellen Sie eine Klasse `CounterExt`, deren Objekte die Aufgabe haben, einen gemeinsamen ZŠhler (Klassenvariable) bis zu einer bestimmten Obergrenze (Festgesetzt Ÿber Parameter des Konstruktors) in verschiedenen Threads hoch zu zŠhlen. Das ZŠhlen wird natŸrlich in der Methode `run()` implementiert. ZŠhlen Sie zusŠtzlich auch in dem Thread die Anzahl an Schleifenwiederholungen mit (Variable `myCount`). Jeder Thread soll einen Namen zur Identifikation erhalten, der ebenfalls Ÿber den Konstruktor gesetzt werden kann.

Starten Sie dann 3 Threads (zB Tick, Trick und Track) und teilen Sie die ZŠhlarbeit somit auf.

Sobald der ZŠhlwert Ÿberschritten wurde soll folgendes ausgegeben werden: ThreadName AnzahlDurchgŠnge GlobalerZŠhlwert Dauer(ms)

!

Deklarieren Sie die Variable `myCount` mit dem SchlŸsselwort `volatile`. Dies bewirkt dass die Variable immer direkt im Hauptspeicher gelesen/geschrieben wird und nicht Ÿber den CPU Cache.

!

Nutzen Sie fŸr die Ermittlung der Dauer die Klassen `Instant` (`Instant.now()`) und `Duration` (`Duration.between(beginn, ende)`)

''

Sehen Sie sich das Ergebnis genau an. Eigentlich wŸrden wir erwarten, dass die Summe der SchleifendurchlŠufe aller Threads dem Zielwert entspricht. Warum ist das nicht der Fall?

## Schritt 2

Wandeln Sie Ihr Programm dahingehend ab, dass anstatt der Vererbung das Interface `Runnable` zum Einsatz kommt.

## Schritt 3

Experimentieren Sie ein wenig mit den Threads:

- Ÿ Was passiert bei sehr geringer Obergrenze (zB 50)?
- Ÿ Was passiert bei sehr hoher Obergrenze?
- Ÿ Erspart man sich in dem Beispiel Zeit durch die Parallelisierung?

