

## JCStress (Java Concurrency Stress Tests)

In diesem Arbeitsblatt verwenden Sie JCStress [1], das Concurrency Testing Tool des Open JDK, um selten auftretende Concurrency Phänomene zu beobachten.

### 1. JMM Prüfungsaufgabe

Folgender Code war eine Prüfungsaufgabe im FS15:

```
public class JMM {
    private AtomicInteger ai = new AtomicInteger(5);
    private int i = 1;

    public void run() {
        new Thread(() -> {
            i++;
            ai.set(i);
        }, "T1").start();

        new Thread(() -> {
            int _i = i;           // (1)
            int _ai = ai.get();   // (2)
            System.out.println(_i + " " + _ai);
        }, "T2").start();
    }

    public static void main(String[] args) { new JMM().run(); }
}
```

Es werden zwei Threads gestartet und die Frage ist nach den möglichen Ausgaben von T2.

#### Aufgaben

- Überlegen Sie sich die möglichen Ausgaben für das obige Programm.
- Übersetzen Sie das Programm in einen JCStressTest mit dem Klassennamen **JMMTest**. Deklarieren Sie die erwarteten Ausgaben in Form von `@Expected` Annotationen.  
Hinweise:
  - Sie finden die obige Klasse im `jcstress` Package.
  - Auch als `@Actor` annotierte Methoden können einen `II_Result` Parameter haben.
- Führen Sie den Test aus und kontrollieren Sie die Resultate (LE\_Testing/results).  
Um den Test auszuführen führen Sie folgendes Gradle Kommando aus:  
OSX/Linux: `./gradlew jcsJMMTest`  
Win: `gradlew.exe jcsJMMTest`
- [Optional] Vertauschen Sie, wie in der ursprünglichen Prüfungsaufgabe beschrieben, die Zeilen (1) und (2) und führen Sie den Test nochmals aus. Überprüfen Sie ob Ihre Erwartungen und die neuen Testresultate zusammenpassen.

[1] <https://github.com/openjdk/jcstress>

## 2. Counter Test

In dieser zweiten Aufgabe untersuchen wir, welcher minimale Wert beim Ausführen der Klasse JCStressCounter auftritt.

```
@JCStressTest
@Description("Tests concurrent increments.")
@Outcome(id = {"10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20"},
        expect = Expect.ACCEPTABLE_INTERESTING, desc = "Legal interleavings")
@State
public class JCStressCounter {
    private volatile int cnt = 0;

    @Actor
    public void thread1() {
        for(int i = 0; i < 10; i++) {
            cnt++;
        }
    }

    @Actor
    public void thread2() {
        for(int i = 0; i < 10; i++) {
            cnt++;
        }
    }

    @Arbiter
    public void observe(I_Result res) {
        res.r1 = cnt;
    }
}
```

Diese obige Klasse ist ebenfalls im Package jcstress zu finden.

### Aufgaben

- a) Lassen Sie den JCStressCounter Test laufen.  
OSX/Linux: ./gradlew jcsCounter  
Win: gradlew.exe jcsCounter
- b) Schauen Sie sich unter LE\_Testing/results den Report an. Welche Werte sind aufgetreten?  
Passen Sie das @Outcome an und notieren Sie den kleinsten Wert, den Sie beobachtet haben.