Threads & Tools

Im Unterricht haben Sie gesehen, wie neue Threads gestartet werden. In diesem Arbeitsblatt werden wir ein Programm mit mehreren Threads zur Laufzeit im Debugger etwas genauer unter die Lupe nehmen. Ziel dieses Arbeitsblattes ist, dass Sie ein besseres Gefühl für Threads bekommen und diese auch debuggen können.

Aufgaben:

1. Laden Sie das 01_Threads.zip Archiv auf Ihren Computer, entpacken Sie es und öffnen Sie es in Ihrer Entwicklungsumgebung.

   Sie finden darin im Paket `worksheet` die folgende Klasse:

```
public class DebugMe {
    public static void main(String[] args) throws Exception {
        Thread t1 = new Thread(new R(1));
        Thread t2 = new Thread(new R(2));
        t1.start();
        t2.start();
        System.out.println("#CPUs: " +
                           Runtime.getRuntime().availableProcessors());
        System.out.println("main: done");
    }
}

class R implements Runnable {
    private int nr;

    public R(int nr) { this.nr = nr; }

    @Override
    public void run() {
       for(int i = 0; i < 100; i++) {
          System.out.println("Hello" + nr + " " + i);
       }
    }
}
```

   Falls beim Ausführen einer Klasse in Eclipse oder VSCode folgender Fehler auftritt:

```
Error occurred during initialization of boot layer
java.lang.module.FindException: Module javafx.controls not found, required by
threads
```

   Führen Sie in der Konsole folgendes Kommando aus:
   `WIN> gradlew.bat eclipse` `Linux/OSX> ./gradlew eclipse`

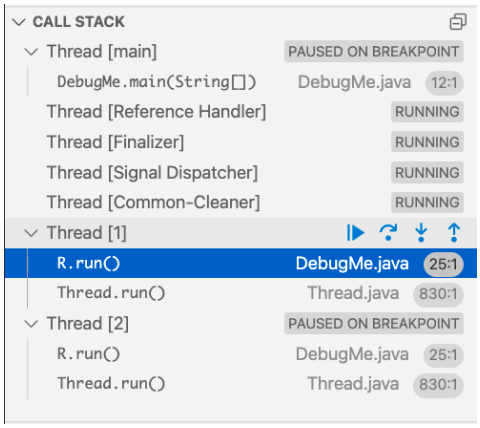
   Dabei werden die Metadaten für die IDE so generiert, dass die Javafx Module korrekt gefunden werden.

2. Führen Sie das Programm mehrmals hintereinander aus. Sehen Sie immer die gleichen Resultate? Wie viele Prozessoren meldet Ihnen die JVM Runtime?

3. Setzen Sie zwei Breakpoints:
   a. Auf der letzten Zeile der main Methode
   b. Innerhalb des For-Loops

   Starten Sie dann das Programm im Debug Mode.

   In der CALL STACK View sehen Sie die drei von Ihnen erzeugten Threads (Thread [1], Thread [2] und Thread [main]) und deren Callstacks. Zusätzlich werden auch die System Threads angezeigt.



   Sie können nun jeden Thread mit den blauen Pfeil-Buttons einzeln kontrollieren. Steppen Sie den Thread 1 in die println Methode und beobachten Sie wie der Callstack wächst. In der Variables View können Sie die Variablen des aktuellen Stackframes und die des aktiven Objektes inspizieren. Primitive Werte (z.B. den int i) können Sie sogar modifizieren.
   Steppen Sie nun so durch das Programm, dass folgender Output ausgegeben wird:

```
#CPUs: n
Thread2: 0
Thread1: 0
Thread2: 1
Thread1: 1
main: done
...
```

4. Erstellen Sie ein Programm, das empirisch die maximale Anzahl Threads ermittelt welche gleichzeitig laufen können. Erzeugen und starten Sie dazu in einer Schleife Threads und geben Sie einen Zähler auf der Konsole aus. Die Threads können Sie z.B. mit der Anweisung `Thread.sleep(Long.MAX_VALUE);` beschäftigen.
   Wie viele Threads können Sie auf Ihrem System starten?

   Warnung:
   Entweder bricht ihr System nach einer gewissen Anzahl Threads mit einer Exception ab, oder aber das System wird instabil. Sichern sie daher alle offenen Dokumente bevor Sie diesen Test ausführen.