Parallel Mergesort mit Executors

In diesem Arbeitsblatt untersuchen Sie eine parallele Mergesort Implementierung. Laden Sie dazu das 08_LE_Tasks.zip Archiv als Eclipse Projekt in Ihren Workspace. Sie finden im Package ws die Klasse MS1, die das Runnable Interface implementiert. Die run Methode ist wie folgt implementiert:

```
public void run() {
    if(end - start <= 1) {            // (a)
        return;
    } else {                          // (b)
        int mid = (start + end) / 2;  // (c)

        MS1 left = new MS1(elems, temp, start, mid, ex);  // (d)
        MS1 right = new MS1(elems, temp, mid, end, ex);   // (e)

        Future<?> lf = ex.submit(left);                   // (f)
        Future<?> rf = ex.submit(right);                  // (g)
         try {
             lf.get(); rf.get();                          // (h)
        } catch (Execution e) {}
        merge(elems, temp, start, mid, end);              // (i)
    }
}
```

Beschreibung:
Wenn der Teilarray ein oder kein Element enthält (a), ist er bereits sortiert und der Task hat sich erledigt. Falls der Array grösser ist (b), wird die Mitte des Arrays berechnet (c). Dann werden zwei Subtasks erstellt - left für die linke Hälfte (d) und right für die rechte Hälfte (e). Beide Subtasks werden dem Executor übergeben und zurück kommen Futures (f)(g). Die zwei Subtasks werden nun parallel abgearbeitet. Dann wartet man auf das Ende der zwei Subtasks (h). Wenn die Subtasks schliesslich erledigt sind, können die beiden Hälften kombiniert werden (i).

Aufgaben:

1. In einer ersten Konfiguration verwenden Sie einen Executors.newFixedThreadPool(3) um einen Array mit vier Elementen zu sortieren. Lassen Sie die main Methode laufen. Kommt das richtige Resultat raus? Kommt überhaupt ein Resultat raus? Finden Sie heraus was schief läuft.
   Hinweis: Falls Sie nicht weiter kommen, kommentieren Sie die zwei print Statements ein.

2. In einer zweiten Konfiguration verwenden Sie einen Executors.newCachedThreadPool(..) der zusätzlich ausgibt, wie viele Threads benötigt werden um einen Array mit vier Elementen zu sortieren. Wie viele Threads sind das? Wie viele Threads sind minimal notwendig?

3. Ist es notwendig, dass beide Subtasks in einem neuen Thread bearbeitet werden? Wie könnte man das System verbessern, bzw. weniger Thread-hungrig gestalten?