

CountDownLatch

Ein Thread soll warten und erst weiterlaufen, wenn ein anderer Thread seine Arbeit getan hat. Das ist ein übliches Szenario in der Programmierung mit Threads und kann mit der Klasse `CountDownLatch` realisiert werden.

In diesem Arbeitsblatt implementieren Sie ein Restaurant mit drei Parteien. Sie haben einen Koch, mehrere Kunden und einen Abwascher. Alle drei Klassen sind als Thread implementiert. Der Koch bereitet für alle Gäste in einer grossen Pfanne dasselbe Menü zu. Zugeben, es ist ein eigenartiges Restaurant =).

Die Abhängigkeiten der Abläufe sind nun wie folgt:

- Die Gäste können erst essen, wenn der Koch das Essen fertiggekocht hat.
- Der Abwascher darf erst mit dem Abwasch beginnen, wenn die Gäste fertig gegessen haben.

Eine unvollständige Implementierung des Restaurants finden Sie im `LE_Synchronizers.zip` Archiv im `latch` Package. Hier sehen Sie die `main`-Methode, in der die drei Parteien erzeugt und gestartet werden:

```
public static void main(String[] args) {
    int nrGuests = 2;

    /*TODO Create latches. */

    new Cook(/*TODO */).start();

    for(int i = 0; i < nrGuests; i++) {
        new Guest(/*TODO */).start();
    }

    new DishWasher(/*TODO */).start();
}
```

Aufgaben

1. Lassen Sie das Programm laufen und beachten Sie, dass der Abwascher schon abgewaschen hat, bevor der Gast sein Menü vertilgt hat.
2. Lesen Sie die JavaDoc des `j.u.c.CountDownLatch` und verwenden Sie dieses um die Abläufe in eine geordnete Reihenfolge zu bringen. Hinweis: Sie benötigen mehr als ein Latch und müssen diese über die Konstruktoren übergeben.
3. Was passiert wenn ein Gast ins Restaurant eintritt nachdem der Koch bereits fertig gekocht hat?
4. (Optional) Erweitern Sie das Restaurant so, dass mehr Köche Ihre Speise fertig gekocht haben müssen, bevor ein Gast essen kann.