BlockingQueue

In diesem Arbeitsblatt implementieren Sie eine Order-Processing-Pipeline. Das System besteht aus drei Komponenten. Kunden, Validierung und Processing. Kunden erzeugen Bestellungen und geben Sie auf. Die Validierer überprüfen die Bestellungen bezüglich Konsistenz und werfen ungültige Bestellungen weg. Processors führen dann die validierten Bestellungen aus. Alle drei Komponenten sind als separate Threads implementiert.

Eine zentrale Eigenschaft des Systems ist, dass sich die Komponenten gegenseitig nicht direkt kennen, sondern über generische Datenstrukturen (BlockingQueues) miteinander kommunizieren. Dieses Muster nennt man Producer-Consumer-Pattern. Eine unvollständige Implementierung des Systems finden Sie im LE_Synchronizers.zip Archiv im blockingQueue Package. In der main Methode werden die einzelnen aktiven Komponenten gestartet:

```
public static void main(String[] args) {
    int nCustomers = 10;
    int nValidators = 2;
    int nProcessors = 3;
    /*TODO: Create Queues */

    for(int i = 0; i < nCustomers; i++) {
        new Customer(""+i, /*TODO */).start();
    }

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

    for(int i = 0; i < nProcessors; i++) {
        new OrderProcessor(/*TODO */).start();
    }
}
```

Aufgaben:

1. Zeichnen Sie eine Skizze des Systems.
2. Vervollständigen Sie das System, indem Sie die einzelnen Komponenten mittels BlockingQueues verbinden. Lesen Sie dazu die JavaDoc der `j.u.c.BlockingQueue` und suchen Sie sich eine geeignete Implementierung.
3. (Optional) Bauen Sie das System so um, dass ein Kunde maximal drei Sekunden wartet, bis seine Bestellung angenommen wird. Sonst wird die Bestellung verworfen und mit einer neuen weitergemacht.