

## Programmierung 2 - Sommersemester 2020

Prof. Dr. Peter Birkner

### Übungsblatt Nr. 20 Abgabe KW 27

**Allgemeiner Hinweis:** Implementieren Sie für alle Ihre Lösungen einen Dialog-Test, auch wenn dies bei einzelnen Aufgaben nicht explizit in der Aufgabenstellung genannt ist. Der Test sollte so gestaltet sein, dass die Funktionsweise Ihres Programms bei der Abnahme gut nachvollzogen werden kann.

#### 1. Aufgabe

Implementieren Sie ein Programm mit einer Producer- und einer Consumer-Klasse. Die Klassen sollen `Producer` und `Consumer` heißen. Der Producer erzeugt zufällige Integer, welche in einer geeigneten Collection gespeichert werden. Der Consumer entnimmt die Integer aus der Collection und berechnet die Quersumme. In einer Schleife sollen Producer und Consumer in einer zufälligen Reihenfolge aufgerufen werden. Etwa wie folgt:

```
Random ran = new Random();
for(int i = 0; i<10000; i++)
{
    if(ran.nextInt(2) > 0 )
        // Erzeugen eines neuen Integers durch den Producer und speichern
        // in einer Collection
    else
        // Entnehmen eines Integeres aus der Collection und Berechnung der
        // Quersumme durch den Consumer
}
```

Beachten Sie bei der Implementierung folgende Aspekte:

- (a) Die Entnahme der Integer aus der Collection soll entweder nach FIFO-Ordnung oder entsprechend der natürlichen Sortierung der Integer erfolgen. Welche der beiden Varianten genutzt wird, soll beim Programmstart als Parameter übergeben werden. Überlegen Sie, welche Collection für die jeweilige Variante am besten geeignet ist.
- (b) Die Producer-Klasse besitzt eine Methode `int produce()` welche einen zufälligen Integer zwischen 0 und 1000 erzeugt und zurückgibt.
- (c) Die Consumer-Klasse besitzt folgende Methoden:
  - `void consume(int i)`: Diese Methode nimmt einen Integer entgegen und berechnet die Quersumme. Für jede Berechnung soll außerdem der Zeitstempel der Berechnung gespeichert werden. Nutzen Sie die Methode `System.currentTimeMillis()`, um den Zeitstempel zu erhalten.

Die Zeitstempel müssen so gespeichert werden, dass für eine gegebene Quersumme alle zugehörigen Zeitstempel effizient zugegriffen werden können. Zugehörige Zeitstempel heißt: Die Zeitstempel aller Berechnungen, die zu der gegebenen Quersumme geführt haben. Es soll außerdem möglich sein effizient zu zählen, wie häufig eine gegebene Quersumme berechnet wurde. Überlegen Sie welche Datenstruktur dies am besten unterstützt. Hinweis: Sie müssen eine Map und eine Collection in geeigneter Weise miteinander kombinieren.

- `int numberOfDifferentResults()`: gibt an, wie viele unterschiedliche Quersummen berechnet wurden
- `int numberOfOccurrences(int i)`: gibt für einen gegebenen Integer an, wie häufig dieser als Ergebnis einer Berechnung vorkam
- `Collection<Integer> getCrossTotalsAscending()`: gibt eine Collection zurück, welche die berechneten Quersummen in aufsteigender Reihenfolge enthält
- `Collection<Integer> getCrossTotalsDescending()`: gibt eine Collection zurück, welche die berechneten Quersummen in absteigender Reihenfolge enthält.
- `Collection<Long> getTimestampsForResult(int i)`: nimmt einen Integer entgegen und gibt eine Collection zurück, welche alle zugehörigen Zeitstempel enthält. D.h. die Zeitstempel der Berechnungen, die zu dem gegebenen Ergebnis geführt haben.

Implementieren Sie eine Testklasse, um Ihre Implementierung zu testen.

**Hinweis:** Ein wichtiger Aspekt der Bewertung ihrer Lösung ist, ob Sie geeignete Datenstrukturen gewählt und diese in geeigneter Weise genutzt haben. Nutzen Sie die Funktionalität der Java-Collections, um die Anforderungen umzusetzen; z.B., um die Zeitstempel in aufsteigender Reihenfolge zu erhalten.

Sie müssen jetzt die Java-Collections verwenden, da alle die Methoden eine Collection zurückgeben.

2. Aufgabe Ändern Sie die `Lager`-Klasse aus Übungsblatt 18 so, dass Sie zur Speicherung der Artikel eine `java.util.Map<K,V>` anstelle eines Arrays verwenden. Wählen Sie Ihre Datenstruktur so, dass Sie anhand der Artikelnummer in konstanter Zeit auf die Artikel zugreifen können. Außerdem sollte es möglich sein, in der Reihenfolge, in der die Artikel zum Lager hinzugefügt wurden, über die Artikel zu iterieren.

Aufgrund der Umstellung auf eine Map werden einige Methoden, wie z.B. `erweitere_lager()`, nicht mehr benötigt, diese können Sie entfernen.