PM2 Java: Collections und Streams Bibliothek

Quelltext: v7 Collections und Streams Bibliothek

AUFGABENSTELLUNG

Teil A

- Eine Bibliothek verwaltet ihre Bücher in einem Bestand (Datenstruktur ***Map***), der unter den Autoren die Bücher der Autoren ablegt.
- Bücher haben einen Autor, einen Titel, ein Erscheinungsjahr und eine ISBN-Nummer.
- Die Klasse ***Buch***, die Klasse ***Bibliothek*** und einige Methoden der Klasse ***Bibliothek*** sind bereits gegeben.
- Sie sollen die Klasse ***Bibliothek*** um Methoden / Funktionalitäten erweitern:
  1. Methode ***add(buch)***: Trägt unter dem Namen des Autors von ***buch*** ein weiteres Buch ***buch*** ein. Dabei muss sichergestellt werden, dass keine doppelten Bücher eingetragen werden. (Sie müssen keine Änderungen in Buch vornehmen!)
  2. Machen Sie die Bibliothek iterierbar.

Teil B

- Implementieren Sie die nachfolgenden Methoden für die Klasse Bibliothek wenn immer möglich mit den Methoden des Java Streaming API's

1. Methode ***produktivsterAutor()***: Berechnet den Autor, der die meisten Bücher geschrieben hat. Wenn die Bibliothek keine Bücher enthält, dann ist das Ergebnis ***null***.
2. Methode ***alleBuecher()***: Gibt alle Bücher der Bibliothek als Menge zurück.
3. Methode ***erschienenNachJahr(jahr)***: Gibt alle Bücher zurück, die nach dem Jahr ***jahr*** erschienen sind.
4. Methode ***inIsbnTabelle()***: Wandelt die Bestand der Bibliothek in eine Tabelle mit Paaren aus ISBN-Nummer und Buch um.
5. Methode ***gruppiereNachJahr()***: Gruppiert alle Bücher des Bestandes nach Erscheinungsjahr. Das Ergebnis ist ein Hash, dessen Schlüssel die Jahre und dessen Werte Mengen von Büchern sind.
6. Methode ***sortiereNachAutorMitTitel()***: Sortiert den Bestand der Bibliothek nach Autoren und sortiert auch die Bücher eines Autors nach Titel.

Hinweise zu B.1

- Das Kriterium für die Bestimmung des Maximums ist über den Wert der ***Map*** definiert. Gesucht ist allerdings der dazu gehörige Schlüssel ➔ Bei der Bestimmung des Maximums darf die Beziehung zwischen Schlüssel und Wert nicht verloren gehen.

- Von welchem Typ ist das Ergebnis von ***max***?

- Wie müssen Sie mit dem Ergebnis verfahren, so dass Sie einerseits den Schlüssel lesen können und andererseits verhindern, dass keine ***NullpointerException*** zur Laufzeit generiert wird?

Hinweise zu B.2 und B.4

- **B.2**: Um einen Strom von Mengen in einen flachen Strom der Elemente der Mengen (ohne Dubletten) zu verwandeln, verwenden wir die Methoden ***flatMap(Set::stream)***. Dies ist eine Art „**flatten“** für Ströme und ist mit ***List***-Typen als Argument verwendbar.

- **B.4**: Um einen Strom von Elementen in eine ***Map*** zu transformieren, verwenden wir die Methode ***Collectors.toMap*** mit zwei Lambda-Ausdrücken als Argumente. Das 1‘te Argument beschreibt wie ein Schlüssel, das 2‘te wie der Wert der Tabelle aus dem Element des Stroms berechnet wird.

Hinweise zu B5 und B6

- **B5.**:Um einen Strom von Elementen in eine ***Map*** mit einem Kriterium für das Bilden von Gruppen zu überführen, verwenden wir die Methode ***Collectors.groupBy***.

- **B6.**: Um einen Strom von Elementen zuerst nach dem ersten Kriterium und dann nach dem 2'ten Kriterium zu sortieren, verwenden wir die Methode ***Comparator.comparing(<Methoden-Referenz>).thenComparing(<Methoden-Referenz>)***. Diese Muster kann auch für mehr als 2 Kriterien angewendet werden.