Aufgabe 3

Lernziele:

- Komplexere Collection und Map Strukturen verarbeiten
- Anwenden des Java-Streaming API's
- Verwenden von Functional Interfaces

Die Aufgabe muss zu Beginn des Praktikums vollständig bearbeitet sein!

Teil A Auswertungen auf der Comic Film Map aus Aufgabe 1

Implementieren Sie die folgenden Auswertungen für die Comic-`Map` aus Aufgabe 1 mit Hilfe statischer Methoden. Die Methodensignaturen sind in der Projektvorlage bereits vorgegeben.

1. Berechnen Sie eine Liste der Comics.
2. Berechnen Sie eine Menge aller Filme.
3. Berechnen Sie eine Liste aller Comics, die in einem gegebenen Jahres Intervall verfilmt wurden.
4. Berechnen Sie eine Menge aller Filme, die in einem gegebenen Jahres Intervall erstellt wurden.
5. Berechnen Sie eine Liste aller Comics, die mit einem gegebenen Präfix beginnen.
6. Bilden Sie eine `Map`, die Jahres Intervalle auf Verfilmungen abbildet, auf die Gesamtzahl der Verfilmungen ab. Das Ergebnis ist vom Typ `int`.
7. Schreiben Sie eine statische Methode, die eine `ToIntFunction` zurückliefert, um ein Key-Value-Paar der Comic-Film-`Map` auf die Anzahl der Verfilmungen abzubilden. Nutzen Sie die `ToIntFunction` in Aufgabenteil 8.
8. Berechnen Sie einen Comic mit den meisten Verfilmungen.
9. Berechnen Sie alle Comics mit den meisten Verfilmungen.
10. Prüfen Sie, ob es irgendeinen Film gibt, der in einem vorgegebenen Jahres Intervall produziert wurde. Die Methode sollte maximal effizient sein und bei dem ersten Film, der die Bedingung erfüllt, terminieren.
11. Gruppieren Sie die Comics nach Jahresintervall. Das Ergebnis ist eine `Map`, deren Schlüssel die Jahres Intervalle und deren Werte Mengen von Comics sind, für die es zu dem Jahres Intervall eine Verfilmung gibt. Arbeiten Sie mit geschachteltem `forEach` und der Methode `computeIfAbsent`.

Teil B Traversieren und Manipulieren von Binärbäumen mit Functional Interfaces

Implementieren Sie für die generische Klasse `BinaryTree<T extends Comparable<? super T>>` drei Objekt-Methoden:

1. `process`: Die Methode hat als Parameter das Functional Interface `Function`, die die Werte des Baumes transformiert. Der Typ der Ergebnis-Werte kann sich vom Typ der Originalwerte unterscheiden. Die Ergebniswerte sollen in einer Liste eingesammelt werden und als Ergebnis der Methode zurückgegeben werden. Experimentieren Sie mit Binärbäumen, die Elemente unterschiedlichen Typs enthalten und rufen Sie die Methode je mit minimal zwei Transformationsfunktionen auf.
2. `process`: Die Methode hat als Parameter die Functional Interfaces `Function` und `Predicate`, die die Werte des Baumes transformieren und filtern. Der Typ der Ergebnis-Werte kann sich vom Typ der Originalwerte unterscheiden. Die Ergebniswerte sollen in einer Liste eingesammelt werden und als Ergebnis der Methode zurückgegeben werden. Experimentieren Sie mit Binärbäumen, die Elemente unterschiedlichen Typs enthalten und rufen Sie die Methode je mit minimal zwei Transformationsfunktionen und Prädikaten auf.
3. `process`: Die Methode hat als Parameter die Functional Interfaces `Function`, `Predicate` und `Consumer`, die die Werte des Baumes transformieren, filtern und dann konsumieren (z.B. durch Drucken). Der Typ der Ergebnis-Werte der Funktion kann sich vom Typ der Originalwerte unterscheiden. Experimentieren Sie mit Binärbäumen, die Elemente unterschiedlichen Typs enthalten und rufen Sie die Methode je mit minimal zwei Transformationsfunktionen, mit minimal zwei Prädikaten und einem `Consumer` auf.

Die generische Klasse `BinaryTree<T extends Comparable<? super T>>` ist inkl. einer Iterator-Implementierung in der Projektvorlage enthalten. Die Projektvorlage liegt im Teamsraum.