Aufgabenblatt 6

Allgemeine Anmerkungen

Ihre Lösung für dieses Aufgabenblatt ist bis Freitag, 22.5., 13h durch `git commit` und `push` abzugeben. Mit der Angabe werden die Dateien `CelestialSystemIndexTreeVariantC.java` (C steht für Comparator), `CelestialBodyComparator.java` und `CelestialBodyNameComparator.java` sowie `CelestialBodyIterable.java`, `CelestialBodyIterator.java` und `CelestialBodyCollection.java` mitgeliefert. Wenn Sie zusätzlich zu den gefragten Klassen, weitere Klassen oder Interfaces definieren, achten Sie darauf, dass die Klassennamen mit `My` beginnen, um Konflikte mit späteren Aufgabenblättern zu vermeiden.

Ziel

Ziel der Aufgabe ist die Anwendung der Konzepte: Sichtweise vs. Kopien, Iterator, Sortieren (siehe Skriptum Seiten 91-114).

Gegebenes Interface

Gegeben ist das Interface `CelestialBodyComparator`, das eine Vergleichsfunktion spezifiziert, die für Objekte der Klasse `CelestialBody` eine Ordnung definiert. Ein Objekt vom Typ `CelestialBodyComparator` wird beim Instanzieren von `CelestialSystemIndexTreeVariantC` angegeben. Weiters sind Interfaces `CelestialBodyCollection`, `CelestialBodyIterable` und `CelestialBodyIterator` gegeben.

Aufgaben

1. Ändern Sie die Klasse `CelestialSystem` so, dass sie das Interface `CelestialBodyCollection` implementiert. Ein Iterator von `CelestialSystem` iteriert über alle Himmelskörper in der Reihenfolge, in der sie mit `add(CelestialBody)` hinzugefügt wurden.
2. Vervollständigen Sie die Klasse `CelestialBodyNameComparator` als Untertyp von `CelestialBodyComparator`. Instanzieren Sie in `Simulation` die Klasse `CelestialSystemIndexTreeVariantC` und stellen Sie durch entsprechende Testfälle sicher, dass sich die Klasse so verhält, wie in `CelestialSystemIndex` beschrieben. Sie dürfen die Implementierung auch ändern, solange weiterhin ein eigens definierter binärer Suchbaum benutzt wird und der Konstruktor mit einem Parameter vom Typ `CelestialBodyComparator` zur Verfügung steht.
3. Ändern Sie die Klasse `CelestialSystemIndexTreeVariantC` so, dass sie zusätzlich das gegebene Interface `CelestialBodyIterable` implementiert. Bei der Implementierung soll darauf geachtet werden, dass die Reihenfolge der vom Iterator gelieferten Himmelskörper der Ordnung im Suchbaum entspricht.

4. Vervollständigen Sie in der Klasse `CelestialSystemIndexTreeVariantC` die Methode `bodies()`, die ein Objekt vom Typ `CelestialBodyCollection` liefert. Definieren Sie eine neue Klasse, deren Namen Sie selbst wählen können, die eine `CelestialBodyCollection` Sichtweise auf die gespeicherten Schlüssel eines Objekts der Klasse `CelestialSystemIndexTreeVariantC` liefert. Analog zum Listing 3.39 im Skriptum auf Seite 108 soll die Klasse so implementiert sein, dass sich nachträgliche Änderungen im Suchbaum auch im von `bodies()` zurückgelieferten Objekt zeigen. Die Methode `add` von `CelestialBodyCollection` soll in Ihrer neuen Klasse so implementiert werden, dass diese immer `false` liefert, also die Sammlung unverändert lässt.
5. Vervollständigen Sie in der Klasse `CelestialSystemIndexTreeVariantC` die Methode `bodiesAsCelestialSystem()`, die alle Schlüssel als unabhängige Sammlung von Himmelskörpern liefert, das heißt, wenn es zu nachträglichen Änderungen im Suchbaum kommt, wirken sich diese nicht auf das zurückgelieferte Objekt aus.

Zusatzfragen

Bitte beantworten Sie folgende Zusatzfragen als Kommentare in der Datei `CelestialSystemIndexTreeVariantC`.

1. Wie verhalten sich die von `bodies()` und `bodiesAsCelestialSystem()` zurückgelieferten Objekte, wenn deren enthaltene Himmelskörper durch `move` bewegt werden? Werden dadurch die Himmelskörper des Suchbaums geändert? (Anmerkung: diesbezüglich gibt es im diesem Aufgabenblatt keine Vorgaben).
2. Wie verhalten sich Ihre Iteratoren, wenn Objekte geändert werden?