Institut für Informatik
Prof. Dr.-Ing. Elke Pulvermüller
Lars Huning, M.Sc.

Universität Osnabrück, 31.05.2021
Testat bis 09.06.2021, 18:00 Uhr

Übungen zu Einführung in die Software-Entwicklung

Sommersemester 2021

Blatt 7

Aufgabe 7.1: Iterator - Implementierung (22 Punkte)

Erweitern Sie die Klasse `MyList` aus dem mitgelieferten Ordner um das Interface `java.lang.Iterable` und einen entsprechenden *fail-fast* `Iterator`, der nach der Vorgabe des Interfaces `java.util.Iterator` dazu in der Lage ist, die Liste zu durchlaufen und dabei Elemente zu löschen. Die Interfaces finden Sie unter folgenden Links:

- Iterator: `https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Iterator.html`
- Iterable `https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Iterable.html`

Achtung! Um die Korrektur zu erleichtern, sollen die Implementation des `Iterable` Interfaces nicht in in derselben Datei erfolgen, wie bei der Aufgabe „Visitor und Visitable - Implementierung“. Aus diesem Grund ist die Klasse `MyList` auch zweimal dem Aufgabenblatt beigefügt. Einmal im Ordner `iterator` und einmal im Ordner `visitor`, welche jeweils für die respektiven Aufgaben gedacht sind.

Aufgabe 7.2: Iterator - Testen (12 Punkte)

Schreiben Sie für Ihre Lösung der Aufgabe „Iterator - Implementierung“ eine Testklasse, die die Funktionen Ihres `Iterator` automatisiert testet. Dies umfasst das Durchlaufen aller Elemente der Liste mithilfe des Iterators, dass Löschen ein oder mehrerer Elemente, sowie ob unter den gegebenen Umständen die richtigen Exceptions geworfen werden.

Aufgabe 7.3: Visitor und Visitable - Implementierung (20 Punkte)

Betrachten Sie die Interfaces `Visitor` und `Visitable` und machen Sie sich mit deren Funktionsweise vertraut. Jede Klasse, die das Interface `Visitable` implementiert, soll beim Aufruf der Methode `accept(Visitor)` all ihre Elemente durchlaufen und für jedes Element die Methode `visit(Object)` der übergebenen `Visitor`-Instanz aufrufen. Dies wird so lange gemacht, bis entweder alle Elemente durchlaufen wurden, oder bis der `Visitor` `false` zurück liefert. Ein `Visitor` liefert also `true`, solange er noch weitere Elemente besuchen will.

Implementieren Sie das Interface `Visitable` in der Klasse `MyList`, welche Sie im Ordner `visitor` dem Aufgabenblatt beigefügt finden. Das Interface soll dabei so implementiert werden, dass mit einem Aufruf von `accept` die Liste einmal vollständig durchlaufen wird, wenn der `Visitor` dies mit seiner Rückgabe zulässt.

Achtung! Um die Korrektur zu erleichtern, sollen die Implementation des `Visitable` Interfaces nicht in in derselben Datei erfolgen, wie bei der Aufgabe „Iterator - Implementierung“. Aus diesem Grund ist die Klasse `MyList` auch zweimal dem Aufgabenblatt beigefügt. Einmal im Ordner `visitor` und einmal im Ordner `iterator`, welche jeweils für die respektiven Aufgaben gedacht sind.

Aufgabe 7.4: Visitor und Visitable - Testen (12 Punkte)

Testen Sie Ihre Implementierung aus der Aufgabe „Visitor - Implementierung“in einer separaten Testklasse. Hierzu müssen Sie das Interface `Visitor` implementieren. Testen Sie für mindestens zwei verschiedene `Visitor`-Implementierungen. Eine `Visitor`-Implementierung soll dabei durch alle Elemente des `Visitable` laufen, während die andere `Visitor`-Implementierung das Besuchen nach einem bestimmten Element abbricht (das konkrete Abbruchkriterium können Sie frei wählen).

Aufgabe 7.5: Persistentes Array - Implementierung (20 Punkte)

Implementieren Sie eine Wrapper-Klasse mit der `Integer`-Arrays persistent abgespeichert, durchlaufen und ihre Einträge verändert werden können. Eine Instanz dieser Klasse soll mit einem `Integer`-Array und einem Namen, unter dem das Array als Datei abgespeichert werden soll, instanziiert werden können. Existiert unter dem Namen bereits eine Datei, soll diese überschrieben werden. Alle Array-Einträge werden dann in die Datei geschrieben. Es soll auch möglich sein, auf ein bereits existierendes, persistentes Array durch Instanziierung der Wrapper-Klasse nur unter Angabe des richtigen Dateinamens Zugriff zu erlangen. Mit einer Instanz schließlich soll man die einzelnen Einträge einsehen und verändern können. Alle Änderungen sollen sofort persistent in die Datei geschrieben werden. Achten Sie darauf, das man auch die Anzahl der Einträge erfragen und die Datei explizit schließen kann.

Hinweis: Sie brauchen nicht mit Streams arbeiten.

Aufgabe 7.6: Persistentes Array - Test (14 Punkte)

Schreiben Sie für Ihre Lösung aus der Aufgabe „Persistentes Array - Implementierung“ eine Testklasse, die automatisiert die von Ihnen implementierten Funktionen testet. Testen Sie auch darauf, ob die von Ihnen angekündigten Exceptions korrekt geworfen werden.

Hinweis: Sie brauchen nicht mit Streams arbeiten.