

Praktikumstutorium – Blatt 7

Als Vorbereitung auf das Testat 7 sollten Sie unbedingt diese Aufgaben bearbeiten.

Sofern dabei Schwierigkeiten auftreten, sollten Sie unbedingt das **Tutorium des Programmierpraktikums** besuchen

Tutoriumszeiten im Raum OH 12/4.030

Montag	Dienstag	Mittwoch	Donnerstag	Freitag
12.00 – 14.00 Uhr	10.00 – 16.00 Uhr	10.00 – 14.00 Uhr	10.00 – 14.00 Uhr	Testat

Strategien für die vorgegebene Klasse `DoublyLinkedList<T>`

Die Grundlage für diese Aufgaben bildet eine Version der Klasse `DoublyLinkedList<T>`, die abstrakten Klassen der aus der Vorlesung bekannten Strategien und die entsprechenden Fassungen der Durchläufe zum Einsatz von passenden Strategie-Objekten enthält. Diese Version der Klasse `DoublyLinkedList<T>` können Sie im **moodle**-Kursbereich im Ordner *Praktikumsaufgaben* herunterladen.

Erweitern Sie die in der Klasse `Testumgebung` vorgegebene Testmethode schrittweise so, dass die Korrektheit der nachfolgend beschriebenen Strategien überprüft wird.

Alle Strategien 1 bis 14 sollen ausschließlich auf Listen von `Integer`-Objekten arbeiten, also Instanzen der Form `DoublyLinkedList<Integer>`

Die Strategien 1 bis 10 sollen *ohne* Änderungen an der Klasse `DoublyLinkedList<T>` implementiert werden. Nutzen Sie eine der drei in der Klasse `DoublyLinkedList<T>` bereits verfügbaren Methoden zum Anwenden der Strategie auf alle Elemente einer Liste.

Verwenden Sie bei der Bearbeitung weder anonyme Klassen noch Lambda-Ausdrücke.

1 - Strategie `CountXStrategy`

Die Strategie `CountXStrategy` soll zählen, wie häufig ein **int**-Wert `x` (in einer Liste von `Integer`-Objekten) vorkommt, der bei der Erzeugung eines Strategie-Objekts angegeben werden soll. (*Hinweis:* Diese Strategie wird als Beispiel für eine Lösung bereits mit der Aufgabenstellung bereitgestellt.)

2 - Strategie `CountInIntervalStrategy`

Die Strategie `CountInIntervalStrategy` soll zählen, wie häufig die **int**-Werte aus einem (geschlossenen) Intervall `[bottom, top]` in einer Liste vorkommen.

3 - Strategie `AverageOfPositivesStrategy`

Die Strategie `AverageOfPositivesStrategy` soll den Mittelwert aller positiven **int**-Werte einer Liste als **double**-Wert bestimmen. (*Hinweis:* Kommt ein positiver Wert mehrfach in der Liste vor, so soll er auch mehrfach in die Durchschnittsberechnung eingehen.)

4 - Strategie `AllToAbsStrategy`

Die Strategie `AllToAbsStrategy` soll in einer Liste alle **int**-Werte auf ihren Absolutbetrag setzen.

5 - Strategie `AddNToPositivesStrategy`

Die Strategie `AddNToPositivesStrategy` soll in einer Liste alle positiven **int**-Werte um einen Wert `n` erhöhen.

6 - Strategie `DoubleAllInIntervalStrategy`

Die Strategie `DoubleAllInIntervalStrategy` soll in einer Liste alle **int**-Werte aus einem (geschlossenen) Intervall `[bottom, top]` verdoppeln.

7 - Strategie RemoveAllNegativesStrategy

Die Strategie RemoveAllNegativesStrategy soll alle Elemente mit negativen **int**-Werten aus einer Liste löschen.

8 - Strategie RemoveAllInIntervalStrategy

Die Strategie RemoveAllInIntervalStrategy soll alle Elemente mit einem **int**-Wert aus einem (geschlossenen) Intervall [bottom, top] aus einer Liste löschen.

9 - Strategie RemoveAndCountAllInIntervalStrategy

Die Strategie RemoveAndCountAllInIntervalStrategy soll alle Elemente mit einem **int**-Wert aus einem (geschlossenen) Intervall [bottom, top] aus einer Liste löschen. *Zusätzlich* soll die Strategie zählen, wie viele Elemente gelöscht worden sind.

10 - Strategie RemoveSmallerThanPredecessorStrategy

Die Strategie RemoveSmallerThanPredecessorStrategy soll alle Elemente aus einer Liste löschen, deren **int**-Wert – in der Ausgangsliste – kleiner als der **int**-Wert des Vorgängerelements ist. Das erste Element einer Liste hat keinen Vorgänger und bleibt daher immer erhalten.

Klasse InsertionStrategy<E> für die Klasse DoublyLinkedList<T>

Für die Strategien 11 bis 14 soll in der Klasse DoublyLinkedList<T> eine abstrakte Klasse InsertionStrategy<E> ergänzt werden, um eine weitere Form von Algorithmen auf der Liste zu ermöglichen. InsertionStrategy-Objekte sollen zwei Methoden bereitstellen:

```
boolean select( E ref )  
E insert( E ref )
```

Die Strategien sollen durch die Methode insertBehindSelected umgesetzt werden, die folgendermaßen arbeitet:

Die Methode select wird für jedes Element e der Liste aufgerufen und erhält jeweils den Inhalt von e als Argument übergeben. Falls der Aufruf der Methode select für ein Element e den Wert **true** ergibt, soll ein neues Element mit dem von insert gelieferten Objekt **hinter** e eingefügt werden.

Implementieren Sie nun passende Strategien, die Werte in eine Liste mit Integer-Inhalten einfügen.

11 - Strategie OneFollowsZeroStrategy

Die Strategie OneFollowsZeroStrategy soll hinter jedem Auftreten des **int**-Werts 0 in einer Liste ein Element einfügen, das den Wert 1 als Inhalt besitzt.

12 - Strategie SubtotalStrategy

Die Strategie SubtotalStrategy soll hinter jedem Element einer Liste ein Element einfügen, das die Zwischensumme aller vorangehenden **int**-Werte als Inhalt enthält.

13 - Strategie SubtotalOfThreeElementsStrategy

Die Strategie SubtotalOfThreeElementsStrategy soll hinter jedem dritten Element einer Liste ein neues Element einfügen, das die Zwischensumme der drei vorangehenden **int**-Werte als Inhalt enthält. Möglicherweise verbleiben am Ende der Liste Elemente, deren Inhalte nicht in die Berechnung einer Zwischensumme eingehen.

14 - Strategie InsertFromListStrategy

Die Strategie InsertFromListStrategy soll hinter jedem Element einer Liste ein Element einfügen. Die Inhalte der eingefügten Elemente sollen nacheinander einer zweiten Liste entnommen werden, die bei der Erzeugung eines Strategie-Objekts angegeben werden soll. Enthält diese zweite Liste nicht genügend viele Einträge, so soll die Ausführung der insert-Methode mit einer Ausnahme abbrechen.