Übungsblatt 10

Strategien für die Klasse `DoublyLinkedList<T>`

Die Grundlage für diese Aufgaben bildet die Klasse `DoublyLinkedList<T>` mit den Klassen für die aus der Vorlesung bekannten Strategien und den entsprechenden Durchläufe zum Einsatz von passenden Strategie-Objekten enthält.

Die Strategien 1 bis 14 sollen ausschließlich auf Instanzen der Form `DoublyLinkedList<Integer>` arbeiten.

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

Verwende bei der Bearbeitung weder anonyme Klassen noch Lambda-Ausdrücke.

Aufgabe 1 - Strategie `CountXStrategy`

Die Strategie `CountXStrategy` zählt, wie häufig ein `int`-Wert `x` (in einer Liste von `Integer`-Objekten) vorkommt, der bei der Erzeugung eines Strategie-Objekts angegeben werden soll.

Aufgabe 2 - Strategie `CountInIntervalStrategy`

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

Aufgabe 3 - Strategie `AverageOfPositivesStrategy`

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

Aufgabe 4 - Strategie `AllToAbsStrategy`

Die Strategie `AllToAbsStrategy` setzt in einer Liste alle `int`-Werte auf ihren Absolutbetrag.

Aufgabe 5 - Strategie `AddNToPositivesStrategy`

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

Aufgabe 6 - Strategie `DoubleAllInIntervalStrategy`

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

Aufgabe 7 - Strategie `RemoveAllNegativesStrategy`

Die Strategie `RemoveAllNegativesStrategy` löscht alle Elemente mit negativen `int`-Werten aus einer Liste.

Aufgabe 8 - Strategie `RemoveAllInIntervalStrategy`

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

Aufgabe 9 - Strategie `RemoveAndCountAllInIntervalStrategy`

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

Aufgabe 10 - Strategie `RemoveSmallerThanPredecessorStrategy`

Die Strategie `RemoveSmallerThanPredecessorStrategy` löscht alle Elemente aus einer Liste, 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 eine neues Element mit dem von `insert` gelieferten Objekt **hinter** `e` eingefügt werden.

Implementiere nun passende Strategien, die Werte in eine Liste mit `Integer`-Inhalten einfügen.

Aufgabe 11 - Strategie `OneFollowsZeroStrategy`

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

Aufgabe 12 - Strategie `SubtotalStrategy`

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

Aufgabe 13 - Strategie `SubtotalOfThreeElementsStrategy`

Die Strategie `SubtotalOfThreeElementsStrategy` fügt hinter jedem dritten Element einer Liste ein neues Element ein, 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.

Aufgabe 14 - Strategie `InsertFromListStrategy`

Die Strategie `InsertFromListStrategy` fügt hinter jedem Element einer Liste ein Element ein. 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.