Aufgaben zur Vorbereitung auf das Testat 7

Als Vorbereitung auf das Testat 7solltest Du unbedingt diese Aufgaben bearbeiten.

Methoden für doppelt verkettete Listen

Erweitere die aus der Vorlesung bekannte Liste, die mit Objekten der Klassen `DoublyLinkedList` und `Element` realisiert wird.

Beachte, dass Du die Lösungen direkt in der Klasse `DoublyLinkedList` ergänzen sollst. Bei der Bearbeitung der Klausur wirst Du nur die Konstruktoren, die beiden Methoden `isEmpty()`, `size()` und die innere Klasse `Element` nutzen dürfen. Im Testat werden Dir zusätzlich die Methoden `add( Object o )`, `showAll()` und `inspect()` zur Verfügung stehen. Bei der Lösung der Aufgaben darfst Du in der Klasse `DoublyLinkedList` keine zusätzlichen Attribute anlegen.

Testumgebung

Erweitere schrittweise die in der Klasse `Testumgebung` vorgegebene Testmethode. Die Testmethode soll die nachfolgend beschriebenen Methoden aufrufen und geeignete Ausgaben machen, um die Korrektheit der Methoden zu überprüfen.

Ergänze die Klasse `DoublyLinkedList` um folgende Methoden:

1 - Methode `void clear()`

Die Methode `clear()` entfernt alle Elemente aus der Liste.

2 - Methode `Object getLast()`

Die Methode `getLast()` gibt den Inhalt des letzten Elements der Liste zurück. Falls die Liste keine Elemente enthält, wird eine `IllegalStateException` geworfen.

3 - Methode `boolean contains( Object obj )`

Die Methode `contains( Object obj )` gibt `true` zurück, wenn der Inhalt `obj` in den Elementen der Liste vorkommt. Dabei soll die Gleichheit mit der Methode `equals` überprüft werden.

4 - Methode `int count( Object obj )`

Die Methode `count( Object obj )` gibt die Häufigkeit zurück, mit der der Inhalt `obj` in den Elementen der Liste vorkommt. Dabei soll die Gleichheit mit der Methode `equals` überprüft werden.

5 - Methode `boolean allEqual()`

Die Methode `allEqual()` gibt `true` zurück, wenn alle Elemente gleiche Inhalte besitzen. Dabei soll die Gleichheit mit der Methode `equals` überprüft werden.

6 - Methode `boolean containsDouble()`

Die Methode `containsDouble()` gibt `true` zurück, wenn mindestens zwei Elemente gleiche Inhalte besitzen. Dabei soll die Gleichheit mit der Methode `equals` überprüft werden.

7 - Methode `void insert( int n, Object obj )`

Die Methode `insert( int n, Object obj )` fügt ein neues Element mit dem Inhalt `obj` hinter dem Element am Index `n` in die Liste ein. Hat die Liste weniger als `n` Elemente, so wird eine `IndexOutOfBoundsException` geworfen.

8 - Methode `void toArray( Object[] arr )`

Die Methode `toArray( Object[] arr )` trägt in das als Argument an den Parameter `arr` übergebene Feld die Inhalte der ersten `arr.length` Elemente der Liste in der gleichen Reihenfolge ein. Besitzt die Liste weniger Elemente, so sollen die verbleibenden Einträge des Feldes auf `null` verweisen. Die Inhalte der Ausgangsliste sollen nicht kopiert werden, so dass anschließend das Feld und die Liste auf die gleichen Objekte verweisen.

9 - Methode `DoublyLinkedList flip()`

Die Methode `flip()` gibt eine Liste zurück, in der die Inhalte der Liste in umgekehrter Reihenfolge auftreten. Die Inhalte der Ausgangsliste sollen nicht kopiert werden, so dass beide Listen anschließend auf die gleichen Objekte verweisen.

10 - Methode `void remove( int n )`

Die Methode `remove( int n )` löscht das Element am Index `n` der Liste, falls dieses existiert. Der Aufruf `remove(0)` soll also das erste Element löschen, der Aufruf `remove(1)` das zweite Element usw. Beachte die Sonderfälle, dass das einzige, das erste oder das letzte Element gelöscht wird. Hat die Liste weniger als `n+1` Elemente, so wird eine `IndexOutOfBoundsException` geworfen werden.

11 - Methode `void remove( Object obj )`

Die Methode `remove( Object obj )` löscht alle Elemente aus der Liste, die den Inhalt `obj` besitzen. Dabei soll die Gleichheit mit der Methode `equals` überprüft werden. Beachte die Sonderfälle, dass das einzige, das erste oder das letzte Element gelöscht wird. Tritt kein Element mit dem Inhalt `obj` auf, soll nicht geschehen.

12 - Methode `void concat( DoublyLinkedList dll )`

Die Methode `concat( DoublyLinkedList dll )` hängt die als Parameter übergebene Liste an die ausführende Liste an. Die übergebene Liste soll danach leer sein. Erzeuge bei der Implementierung **keine** neuen Objekte der Klasse `Element`.

13 - Konstruktor `DoublyLinkedList( DoublyLinkedList dll )`

Der Konstruktor `DoublyLinkedList( DoublyLinkedList dll )` erzeugt eine Liste, die Kopien der Elemente der Liste `dll` enthält. Die *Inhalte* der Elemente der Ausgangsliste sollen nicht kopiert werden, so dass beide Listen anschließend auf die gleichen Objekte verweisen.

14 - Methode `DoublyLinkedList subList( int from, int to )`

Die Methode `subList( int from, int to )` gibt eine neue Liste mit den Inhalten zurück, die in der Ausgangsliste vom Index `from` (inklusiv) bis zum Index `to` (exklusiv) liegen. Die Ausgangsliste bleibt unverändert. Die Inhalte der Ausgangsliste sollen nicht kopiert werden, so dass beide Listen anschließend auf die gleichen Objekte verweisen. Definiere die Indizes `from` und `to` einen ungültigen Bereich, so soll eine `IndexOutOfBoundsException` geworfen werden.

15 - Methode `void removeAll( DoublyLinkedList dll )`

Die Methode `removeAll( DoublyLinkedList dll )` löscht alle Elemente aus der Liste, die einen Inhalt besitzen, der in der Liste `dll` vorkommt. Dabei soll die Gleichheit mit der Methode `equals` überprüft werden.

16 - Methode `void pack()`

Die Methode `pack` löscht Elemente in der Liste derart, dass von jeder Teilfolge von unmittelbar aufeinander folgenden Elementen mit gleichen Inhalten jeweils nur genau ein Element erhalten bleibt.