Praktikumstutorium – Blatt 6

Als Vorbereitung auf das Testat 6 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*

Methoden für Listen

Erweitern Sie die aus der Vorlesung bekannte Liste, die mit Objekten der Klassen `DoublyLinkedList` und `Element` realisiert wird. Die beiden Klassen `DoublyLinkedList` und `Element` können Sie in der Datei `DAP1-Tutorium-06-Aufgaben.zip` aus dem Ordner `Praktikumsaufgaben` des ***moodle***-Kursbereichs herunterladen.

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

1 - Testumgebung

Erweitern Sie 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.

Ergänzen Sie die Klasse `DoublyLinkedList` um folgende Methoden:

~~2~~ - Methode `void clear()`

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

~~3~~ - Methode `Object getLast()`

Die Methode `getLast()` soll den Inhalt des letzten Elements der Liste zurückgeben. Falls die Liste keine Elemente enthält, soll eine `IllegalStateException` geworfen werden.

~~4~~ - Methode `boolean contains( Object o )`

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

~~5~~ - Methode `int count( Object o )`

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

~~6~~ - Methode `boolean allEqual()`

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

~~7~~ - Methode `boolean containsDouble()`

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

~~8~~ - Methode `void insert( int n, Object o )`

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

~~9~~ - Methode `void toArray( Object[] arr )`

Die Methode `toArray()` soll in das als Argument an den Parameter `arr` übergebene Feld die Inhalte der ersten `arr.length` Elemente der Liste in der gleichen Reihenfolge eintragen. 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.

10 - Methode `DoublyLinkedList flip()`

Die Methode `flip()` soll eine Liste zurückgeben, 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.

11 - Methode `void remove( int n )`

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

12 - Methode `void remove( Object o )`

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

13 - Methode `void concat( DoublyLinkedList dll )`

Die Methode `void concat( DoublyLinkedList dll )` soll die als Parameter übergebene Liste an die aufrufende Liste anhängen. Die übergebene Liste soll danach leer sein. Erzeugen Sie bei der Implementierung **keine** neuen Objekte der Klasse `Element`.

14 - Konstruktor `DoublyLinkedList( DoublyLinkedList dll )`

Der Konstruktor `DoublyLinkedList( DoublyLinkedList dll )` soll eine Liste erzeugen, 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.

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

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

16 - Methode `void removeAll( DoublyLinkedList dll )`

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