Wirtschaftsinformatik
Hochschule Ravensburg-Weingarten

Klausur

Objektorientierte Programmierung Praktikum

Studiengang

Wirtschaftsinformatik und E-Business (Bachelor) WI PLUS

Prof. Dr. Thomas Bayer

Wintersemester 2017

(Bitte in Druckbuchstaben ausfüllen!)

Matrikel-Nr:	Datum: **29.01.2018**
Sitzplatz:	Semester: **WS 2017**
Hilfsmittel: **keine**	Bearbeitungszeit: **120 Minuten**

(Bitte nicht ausfüllen!)

	Unterschrift Prüfer:	Note:

Aufgabe	DEF	Size	Add	Cont	Rem	Forall	Get	Set	rem	Gesamt
max. Punkte	**5**	**10**	**20**	**10**	**10**	**15**	**10**	**10**	**10**	**100**
Punkte										

Ausgangssituation

Das Interface ISimpleList soll mithilfe einer verketteten Liste implementiert werden. Dazu kann die Klasse AbstractSimpleList verwendet werden. In der Liste können nur Objekte aufgenommen werden, die nicht null sind!

Interface ISimpleList

```
public interface ISimpleList<E> {
	int size(); // Anzahl der Elemente in der Liste
	boolean add(E e); // fügt das Element e am Ende der Liste ein,
sofern es ungleich null ist: Liefert true, falls ein Element eingefügt
wurde (false sonst)
	boolean add(Collection<E> c); // fügt alle Elemente der Collection
c ein (sofern diese nicht null sind).
	boolean contains(Object o); // prüft, ob das Objekt o in der Liste
enthalten ist (o == null führt zur Rückgabe false)
	boolean remove(Object o); // entfernt das Objekt o aus der Liste
// true: Ein Element wurde entfernt, false sonst
	void forAll(ICommand<E> command); // Durchläuft alle Elemente der
Liste und wendet command.execute auf jedes Element an

	default void add(int i, E e) throws IndexOutOfBoundsException { … }
// fügt das Element an der Position i ein, sofern es nicht null ist und
0 <= i <= Anzahl an Elementen gilt. Rückgabe true, false sonst.
	default E get(int i) throws IndexOutOfBoundsException { … }
// Liefert das Element, das an Position i steht, null falls i keinen
gültigen Index darstellt.
	default void set(int i, E e) throws IndexOutOfBoundsException { … }
// Ersetzt das i-te Element durch e, sofern es vorhanden ist (Index
gültig)
	default boolean remove(int i) { … } // Löscht das i-te Element,
sofern der Index gültig ist. Rückgabe true, false sonst.
	default void sort(Comparator<E> comparator) { … }
}
```

Interface ICommand

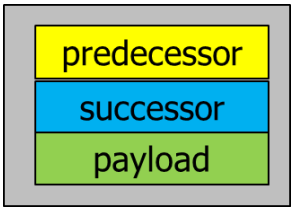
```
public interface ICommand<E> {
	void execute(E e);
}
```

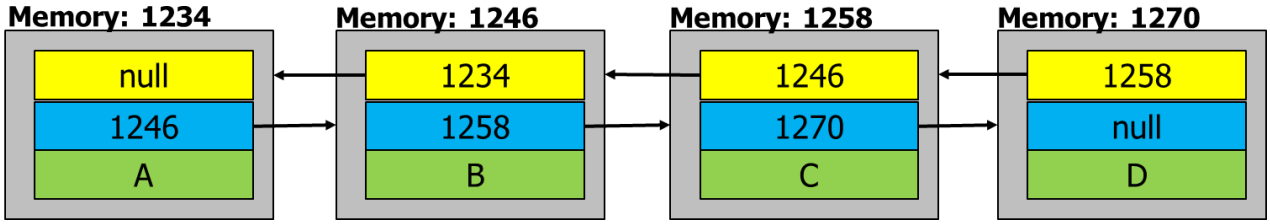
Knoten für die Verkettung

```
public class Node<E> {
	private Node<E> predecessor; // Vorgänger
	private Node<E> successor;   // Nachfolger
	private E payload;           // Nutzlast
…}
```

Hinweise

- Verwenden Sie die Klasse Node<E>, um die verkettet Liste darzustellen.
- Erstellen Sie die Klasse wie oben und fügen Getter/Setter für alle Attribute sowie einen Konstruktor mit Parameter payload hinzu.
- Nutzen Sie hierzu Eclipse!





Hinweise zur Benotung (100 Punkte)

- Programm nicht lauffähig (Syntaxfehler): 0 Punkte
- Collection-Klasse statt doppelter Verkettung: max. 10 Punkte
- Laufzeitfehler: Maximal 30 Punkte
- Sowohl Unit-Tests als auch der Quelltext werden herangezogen

Aufgaben:

- Schreiben Sie Name und Matrikelnummer in den Kopf jeder von Ihnen erstellten Datei
- Erstellen Sie die Klasse SimpleLinkedLists#####<E>, die das Interface ISimpleList implementiert (bspw. durch Erben von der abstrakten Klasse !)
- Programmieren Sie die alle Methoden aus (Bringen Sie die Unit-Tests zum Laufen)

Vorgehen

1. Starten Sie Eclipse mit dem Workspace im Verzeichnis C:\InsightFiles
2. Erstellen Sie das Projekt 2017-WS-Testat-##### (##### = Ihre Matrikelnummer)
3. Importieren Sie das jar-File und JUnit4 (Projekt->Properties)
4. Erstellen Sie das Paket solution#####
5. Erstellen Sie die Klasse SimpleLinkedLists##### (wie oben)
6. Erstellen Sie die Klasse SimpleLinkedListsTest#####, die von der Klasse SimpleListTest90 erbt
7. Implementieren Sie die abstrakten Methoden in beiden Klassen (eclipse verwenden)
8. Bringen Sie die Unit-Tests zum Laufen: Testklasse als JUnit-Test ausführen (fail 23, Error 4)
9. Erstellen Sie die Klasse Node<E>
10. Programmieren Sie die Methoden aus: Starten Sie mit: size, add, contains, forAll, remove, get(i),add(i),set(i)
11. Exportieren Sie Ihr Projekt als Archiv in das Verzeichnis C:\InsightFiles
12. Vor Abgabe melden Sie sich!

Hinweise zur Implementierung

- In der Klasse SimpleLinkedLists##### müssen Sie einen Verweis auf den ersten Knoten (node) speichern
- Eine Schleife über die Knoten sieht schematisch wie folgt aus:
    - node = erster Knoten
    - boolean loop = true;
    - while(loop) {
      //hier kann auf die Nutzlast zugegriffen werden (payolad)
      if(Nachfolger von node == null) loop = false;
      else node = nächster Knoten (der Nachfolger von node!)
      } // node = der letzte Knoten!
    - add: Unterscheiden die Fälle Liste leer / Liste enthält Elemente
- Erstellen Sie eine private Methode, die als Eingabe ein Objekt o und als Rückgabewert Node<E> hat
    - Falls o == null: Der letzte Knoten wird zurückgeliefert
    - Falls o != null: Der Knoten, dessen Nutzlast gleich o ist (o.equals…) wird zurückgegeben. Falls kein Knoten o enthält, wird null zurückgegen.
- Remove: Unterscheiden Sie die folgenden Fälle
    - Liste leer
    - Liste nicht leer: Löschen Sie nicht den Knoten, sondern setzen Sie die payload auf null und reduzieren Sie Anzahl der Elemente um 1
- Indexzugriff: Erstellen Sie eine private Methode, die als Eingabe den Index i und als Rückgabewert Node<E> hat
    - Falls i < 0 oder size >= i, geben Sie null zurück
    - In der Schleife prüfen Sie, ob die payload != null ist. Wenn nein, erhöhen Sie einen Zähler. Danach wird node = Nachfolger ausgeführt. Ist der Nachfolger null, können Sie abbrechen.
- Get(i) / set(i) / remove(i) sind mit der obigen Methode Einzeiler
- Add(i): Unterscheiden Sie folgende Fälle:
    - Size == 0: Falls i != null dann werfen IndexOutOfBoundsException. Sonst (i == size == 0) wird die Nutzlast im ersten Knoten gespeichert
    - Size > 0: Falls i == size, dann fügen Sie das Element am Ende ein (Wiederverwendung!!!)
    - Size > 0, i < size: Positionieren Sie sich auf den Index i (wie oben). Danach müssen Vorgänger und Nachfolger gesetzt werden. Fall kein Vorgänger existiert, handelt es sich um das erste Element (Attribut verwenden)