

Objektorientiertes Programmieren in Objective-C unter Mac OS X

Aufgabe 5 (L) (Objektorientierung und Speichermanagement)

Verwenden Sie als Hilfe zu Ihren kommenden objektorientierten Programmieraufgaben die Dokumentation *The Objective-C Programming Language*. Sie befindet sich in:

`/Developer/Documentation/Cocoa/Conceptual/ObjectiveC/ObjC.pdf`

Insbesondere die ersten 83 Seiten sind von Interesse.

Objective-C bietet zur Speicherverwaltung erweiterte Funktionalität gegenüber C mit Hilfe eines sogenannten *Reference Counters*. Um das Verständnis darüber zu vertiefen, lesen Sie:

`/Developer/Documentation/Cocoa/Conceptual/MemoryMgmt/index.html`

Aufgabe 6 (Doppelt verkettete objektorientierte Listen)

In dieser Aufgabe wird die in Aufgabe 4 auf Aufgabenblatt 2 eingeführte doppelt verkettete Liste objektorientiert implementiert. Durch diese Aufgabe sollen die Unterschiede der imperativen Programmierung zur objektorientierten Programmierung ersichtlich werden.

- (a) Entwerfen sie ein Objekt `TUMLinkedList`, welches die Liste erzeugt und verwaltet.
- (b) Entwerfen Sie ein Objekt `TUMLLElement`, welches ein Element der doppelt verketteten Liste repräsentiert und als Datenelement ebenfalls wieder einen generischen Objektzeiger enthält.
- (c) Fügen Sie zur `TUMLinkedList` folgende Methoden hinzu, die die mit dem Objekt verbundene Liste verwalten:
 - i. - `(id)init`
Initialisiert die Liste, und gibt sich selbst zurück.
 - ii. - `(TUMLLElement *)add:(id)anObject before:(TUMLLElement *)anElement`
Fügt ein Element vor `anElement` in die Liste ein.
 - iii. - `(TUMLLElement *)add:(id)anObject behind:(TUMLLElement *)anElement`
Fügt ein Element nach `anElement` in die Liste ein.
 - iv. - `(TUMLLElement *)add:(id)anObject`
Fügt ein Element ans Ende der Liste an.
 - v. - `(void)swap:(TUMLLElement *)firstElement with:(TUMLLElement *)secondElement`
Vertauscht zwei Elemente.

- vi. - `(void)remove:(TUMLLElement *)anElement`
Entfernt das Element `anElement`.
- vii. - `(TUMLLElement *)first`
Gibt das erste Element der Liste zurück.
- viii. - `(TUMLLElement *)last`
Gibt das letzte Element der Liste zurück.
- ix. - `(void)dealloc`
Gibt den Speicher der gesamten Liste und aller ihrer Elemente frei.

(d) Geben Sie dem Objekt `TUMLLElement` passende Accessor-, `init`- und `dealloc`-Methoden.

(e) Achten Sie auf eine korrekte Speicherverwaltung.

Testattermin: 12.5.2004

Aufgabe 7 (Verkäufer Käufer)

Um die doppelt verkettete Liste aus Aufgabe 6 angemessen zu testen, schreiben sie ein kleines Programm, das einen „Warenaustausch“ zwischen Käufer und Verkäufer simuliert:

- (a) Erstellen Sie jeweils eine Klasse `TUMBuyer` und `TUMSeller` welche in einer doppelt verketteten Liste Ihren Warenbestand speichern und einen „Geldbeutel“ beinhalten.
- (b) Fügen Sie Ihren Klassen sinnvolle `init`-, `dealloc`- und Accessor-Methoden hinzu.
- (c) Führen Sie einen Warenaustausch mit mehreren Käufern und Verkäufern durch.
- (d) Achten sie auf eine korrekte Speicherverwaltung, insbesondere beim Austausch von Objekten.

Testattermin: 12.5.2004