SE1, Aufgabenblatt 13

Softwareentwicklung I – Wintersemester 2012/13

Sortieren, Stack

MIN-CommSy-URL: ..... https://www.mincommsy.uni-hamburg.de/

CommSy-Projektraum ..... SE1 CommSy WiSe 12/13

Ausgabewoche ..... 24. Januar 2013

Kernbegriffe

Sortieren ist eine klassische Problemstellung in der Informatik. Es existieren verschiedene *Sortieralgorithmen* (*Sortierverfahren*) mit unterschiedlicher Laufzeitkomplexität. Sehr häufig müssen Elemente an ihrem aktuellen Speicherort (*in situ* oder *in place*) in eine geordnete Reihenfolge gebracht werden. *Bubble-Sort* ist ein sehr einfaches Sortierverfahren für diese Situation mit quadratischem Aufwand ($O(n^2)$). *Quick-Sort* hingegen gilt als ein sehr schneller Sortieralgorithmus, der im günstigsten Fall eine Komplexität von *O(n log(n))* aufweist, im ungünstigen Fall aber zu $O(n^2)$ degenerieren kann.

Ein *Stack* (im Deutschen auch Stapel oder Kellerspeicher genannt) ist eine Sammlung von Elementen, die nach dem LIFO (Last In First Out) Prinzip verwaltet werden, d.h. zuletzt abgelegte Elemente werden zuerst wieder entnommen (wie etwa bei einem Tablettstapel). Ein Stack definiert im wesentlichen vier Operationen: *push* legt ein Element auf den Stack, *pop* entfernt ein Element vom Stack, mit *peek* kann das oberste Element abgefragt werden, ohne es zu entfernen, und *isEmpty* prüft, ob der Stack leer ist.

Lernziele

Sortierverfahren, die mit Vertauschungen arbeiten, verstehen und auf Basis ihrer abstrakten Beschreibung implementieren können, noch sicherer mit Listen umgehen können, Stacks sowohl implementieren als auch anwenden können.

Aufgabe 13.1 Sortieren einfach

13.1.1 Öffnet das Projekt *Sortieren* und erstellt ein Exemplar der Klasse `VisualIntListe`. Daraufhin öffnet sich ein neues Fenster mit weißen Quadraten. Was bedeuten diese Quadrate? Lest euch die Dokumentation der Klasse durch. Ruft anschließend an dem Exemplar die Operation `initialisiereAufsteigend` auf. Welche Änderungen ergeben sich dadurch, und was bedeuten diese?

13.1.2 Studiert die Schnittstelle `Sortierer` und implementiert anschließend den Bubble-Sort-Algorithmus in der vorgegebenen Klasse `BubbleSortierer`.

13.1.3 Um eure Implementation von `BubbleSortierer` interaktiv zu testen, erzeugt ihr ein Exemplar der Klasse `VisualIntListe` und ein Exemplar der Klasse `BubbleSortierer`. Dann ruft ihr an dem Sortierer die Methode `sortiere` auf und übergebt die zu sortierende Liste als Parameter.

Aufgabe 13.2 Sortieren besser

13.2.1 Implementiert den Quick-Sort-Algorithmus (in place). Vervollständigt dazu die Klasse `QuickSortierer`, welche ebenfalls das Interface `Sortierer` implementiert. Testet wieder interaktiv. Die Teillisten werden im Quelltext durch die beiden Grenzvariablen `von` und `bis` realisiert. Die öffentliche `sortiere`-Methode sorgt dafür, dass der rekursive Prozess mit passenden Werten für die komplette Liste angestoßen wird. Beachtet, dass die Länge der Teilliste von den Grenzvariablen `von` und `bis` abhängt. Es wäre sinnlos, die Operation `gibLaenge` auf der Liste aufzurufen, da diese immer die Länge der kompletten Liste liefert.

13.2.2 *Zusatzaufgabe*: Experimentiere mit verschiedenen Strategien zur Wahl des Pivot-Elements, zum Beispiel erstes Element, mittleres Element, letztes Element, der Median aus diesen drei Elementen, ein zufälliges Element, der Median aus drei zufälligen Elementen, der Median aus 9 zufälligen Elementen…

Aufgabe 13.3 Ein Anwendungsbeispiel für Stacks: Die Postfix-Notation

Ein arithmetischer Ausdruck in Postfix-Notation ist dadurch gekennzeichnet, dass zuerst die Operanden und dann der Operator angegeben werden. Beispiel: „3 7 + 2 /“ würde in der üblichen Infix-Notation dem Ausdruck „(3+7)/2“ entsprechen. Ein postfix-notierter Ausdruck kann immer in der Reihenfolge, in der die Elemente auftreten, ausgewertet werden, die Priorität der Operatoren muss nicht beachtet werden bzw. alle besitzen die gleiche Priorität.

13.3.1. Zum Auswerten eines Ausdrucks in Postfix-Notation kann ein Stack verwendet werden. Dabei wird die Eingabe, bestehend aus Operanden und Operatoren, von links nach rechts eingelesen. Operanden werden auf den Stack gepusht. Beim Auftreten eines Operators holt man sich die obersten beiden Operanden vom Stack, verrechnet diese gemäß dem Operanden, und pusht das Ergebnis wieder auf den Stack.

Verdeutlicht diese Vorgehensweise **grafisch** anhand des Ausdrucks „2 3 * 4 5 * +“.

13.3.2. Öffnet das Projekt *Postfixrechner*. In der Klasse `Postfixrechner` befindet sich bereits eine (bisher leere) Operation zum Auswerten von Postfix-Ausdrücken. Vervollständigt die Implementation.

Ihr könnt in der Implementierung davon ausgehen, dass die Operatoren und Operanden voneinander durch Leerzeichen getrennt sind, und dass außer den Operatoren nur ganzzahlige positive Werte gegeben sind. Benutzt als Stack-Realisierung den Typ `java.util.Stack<Integer>`. Zum Zerlegen der Eingabe in Operatoren und Operanden bietet sich die Operation `split(String regex)` auf Strings an. Um einen String wie „42“ in eine Zahl wie 42 umzuwandeln, gibt es die Operation `Integer.parseInt(String)`.

13.3.3. Welche Fehler können in der Eingabe auftreten, und wie wirken sich diese aus?

Aufgabe 13.4 Stacks testen und implementieren

13.4.1. Öffnet das Projekt *LinkedStack*. Dieses enthält ein Interface `Stack` sowie ein JUnit-Testgerüst. Lest euch die `Stack`-Dokumentation gründlich durch und vervollständigt anschließend das JUnit-Testgerüst. Falls euch weitere Testfälle einfallen, ergänzt die Testklasse entsprechend. Die Tests lassen sich natürlich noch nicht ausführen, da es bisher keine Implementation von `Stack` gibt.

13.4.2. Stacks lassen sich durch Verkettung von Knoten implementieren (vergleiche hierzu `DoppellinkKnoten` und `LinkedTitelListe` in SE1Tunes). Da man einen Stack immer nur am Rand manipuliert, reicht eine einfache Verkettung aus. Überlegt euch, ob es in Bezug auf die Komplexität von `push` und `pop` schlauer ist, ein neues Element „vorne“ oder „hinten“ einzufügen. Fertigt eine **Zeichnung** eines Stacks an, die zeigt, wie die interne Struktur eures Stacks nach dem Einfügen dreier Elemente aussieht.

13.4.3. Implementiert eine Klasse `Node` für die Knoten und eine Klasse `LinkedStack` für den eigentlichen Stack. Passt den Konstruktor der Testklasse an, indem ihr dort ein Exemplar von `LinkedStack` erzeugt.