Selbstanordnende Listen

Tobias Strutz
MNr. 150704

Mein Programm ordnet nach 3 verschiedenen Strategien Elemente in einer Liste.

Nachdem man sich die Datei *FavoriteLists.zip* von
www.tstrutz.de/FavoriteLists.zip
geladen hat, muss sie z.B. mit

> *unzip FavoriteLists.zip*

entpackt werden. Man erhält einen Ordner *FavoriteLists* (Quelltext im Unterverzeichnis *src*). Zu starten ist das Programm im Unterordner *FavoriteLists/bin* mit dem Befehl

> *java beleg/gui/FavoriteLists.*

Es öffnet sich das Hauptfenster (1)

1. Hauptfenster von FavoriteLists

Die Schaltflächen öffnen weitere Fenster in denen die verschiedenen Sortierstrategien realisiert sind.

MTF:

2. Fenster zu Sortierstrategie Move- To- Front

Die Schaltfläche *Init List* initialisiert eine Liste mit 10 verschiedenen Integerzahlen (0- 9), wie im Screenshot (2) zu sehen.
Im Feld darunter kann man nach einer einzugebenden Zahl suchen lassen. Wird diese Zahl in der Liste gefunden leuchtet das Fenster kurz grün auf, und die Liste wird nach der Strategie „Move- To- Front“ neu geordnet. Bei vergeblicher Suche leuchtet das Fenster kurz rot auf.

T:

3. Fenster zu Sortierstrategie Transpose

Die Funktionsweise ist analog zum Fenster (2) mit dem Unterschied, dass hier mit der Strategie „Transpose“ die Liste neu angeordnet wird. Im Screenshot (3) ist eine einmalige Suche nach der Zahl 8 erfolgt.

FC:

4. Fenster zu Sortierstrategie Frequency Count

Die Funktionsweise ist analog zu den beiden vorherigen Fenstern. Sortiert wird mit der Methode „Frequncy Count“. Deshalb ist in der Listenanzeige die Angabe eines Suchzählers enthalten. Diese Zahl gibt die erfolgreichen Suchen zu den jeweiligen Listeneinträgen an.
Z.B. 8|2 – das Element 8 wurde bisher 2 mal gesucht.
Im Screenshot (4) wurde zweimalig nach der 8 und einmal nach der 3 gesucht.

Zum visuellen Vergleich der Ordnungsstrategien können die 3 Fenster gleichzeitig betrieben werden.

Realisierung:

Die 3 Sortierstrategien sind in 3 verschiedenen Klassen implementiert.

MTFNodeList.java
Die Methode *access(Object e)* liefet true zurück, wenn e in der Liste gefunden wird.
Ist das der Fall wird der Knoten des Elementes e aus der Liste gelöscht und an der ersten Position der Liste wieder eingefügt. Diese Funktionalität ist in der Oberklasse *NodeList.java* über deren Methoden *remove(Position)* bzw. *insertFirst(Object e)* realisiert.

TNodeList.java
Diese Klasse erbt auch von *NodeList.java* . Die Funktionsweise ist sehr ähnlich der Klasse *MTFNodeList.java* mit dem Unterschied das die Methode *access (Object e)* den Knoten, mit dem gefundenen Element, mit seinem voranstehenden Knoten vertauscht
(sw*apElements(Position, Position)* von *NodeList.java*).

FCNodeList.java
Diese Klasse erbt nicht von *NodeList.java* , führt aber ein Feld mit einem Objekt von *NodeList.java*. Die Klasse bedient sich ausschließlich der Methode *insertFirst()* seines Feldes um Objekte vom Typ *Entry* in die Liste einzufügen. *Entry*- Objecte beinhalten neben einem Element auch einen Zähler. Entsprechend diesem Zähler wird in der Methode *access(Object e)* ein Knoten des gefundenen Elements nach vorne geschoben (Vergleich mit dem Zähler des vorangehenden Knotens). Das Voranschieben erfolgt so lange bis sich zwei Zähler gleichen oder der Zähler des gefundenen Knotens kleiner ist.