



Vergleich von Datenstrukturen

Florian Pix

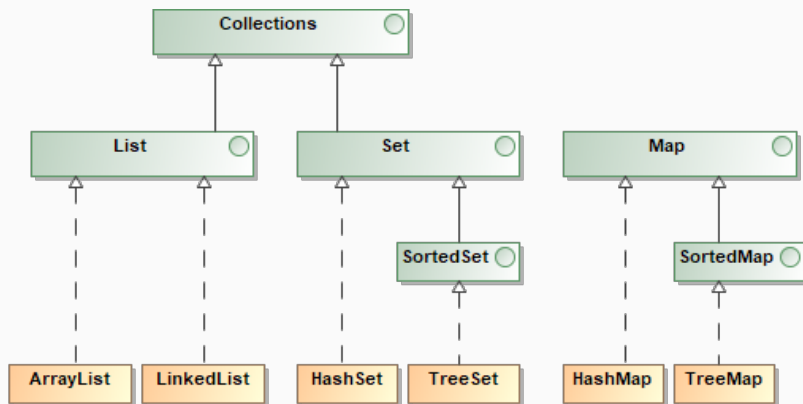
02.05.2019

SWT Übung SoSe19

1. Übersicht
2. List
3. Set
4. Map
5. Vergleich

Übersicht

Übersicht





List

ArrayList




LinkedList

Im Prinzip ein Array mit ein paar Methoden.

- Zugriff auf Elemente per Index 
- Hinzufügen und Entfernen von Elementen langsam 

javadoc

Intern kein Array sondern eine doppelt verkettete Liste.
(Jedes Element hat einen Zeiger auf seinen Vorgänger und Nachfolger.)

- Einfügen und Entfernen von Elementen schnell 
- Zugriff auf einzelnes Element langsam 
- mehr Speicherbedarf 

Set

HashSet

SortedSet

Realisierung der mathematischen Menge.
D.h. es gibt in einem Set keine Dublikate
aber auch keinen Index.

Einfügen und Suchen erfolgt mit Hashs.
Das sind int IDs, die aus den Daten
eines Objekts gebildet werden können.

Hilfe

siehe hashCode()

Mit dem **HashCode** des einzufügenden Elementes sucht HashSet zuerst die Schublade mit dieser Nummer.

Einfügen in HashSets

Mit dem `HashCode` des einzufügenden Elementes sucht `HashSet` zuerst die Schublade mit dieser Nummer.

Wird Sie nicht gefunden, wird das Element als neu eingestuft und in einer neuen Schublade mit dieser Nummer abgelegt.

Falls doch werden alle Objekte in dieser Schublade mit dem neuen Element verglichen.

Dazu verwendet `HashSet` die Methode `equals()`.

Einfügen in HashSets

Mit dem **HashCode** des einzufügenden Elementes sucht HashSet zuerst die Schublade mit dieser Nummer.

Wird Sie nicht gefunden, wird das Element als neu eingestuft und in einer neuen Schublade mit dieser Nummer abgelegt.

Falls doch werden alle Objekte in dieser Schublade mit dem neuen Element verglichen.

Dazu verwendet **HashSet** die Methode **equals()**.

Wird mit dieser Methode kein Objekt gefunden, wird das Objekt in dieser Schublade gespeichert, andernfalls wird es nicht aufgenommen.

Daraus folgt dass wenn wir `equals()` selbst implementieren, man auch die `hashCode()` neu schreiben sollte.
Damit `HashSets` effizient funktionieren.

Daraus folgt dass wenn wir `equals()` selbst implementieren, man auch die `hashCode()` neu schreiben sollte.

Damit `HashSets` effizient funktionieren.

Elemente, die mit `equals` gleich sind,
müssen den gleichen `HashCode` erzeugen.

Ansonsten würden sie in verschiedenen Schubladen gesteckt werden.

Elemente werden sortiert eingefügt nach ihrer Ordnung.
D.h. sie müssen Comparable implementieren.

Map

Besteht aus Schlüssel-Wert-Paaren.
Werte können doppelt auftauchen,
aber Schlüssel müssen einzigartig sein.

- Werte können mit ihrem Schlüssel schnell gefunden werden. 

Vergleich

Vergleich

Data structure	Access	Search	Insert	Delete
List				
ArrayList	$O(1)$	$O(n)$	$O(1)$	$O(n)$
LinkedList	$O(n)$	$O(n)$	$O(1)$	$O(n)$
Map				
HashMap	$O(1)$	$O(1)$	$O(1)$	$O(1)$
LinkedHashMap	$O(1)$	$O(1)$	$O(1)$	$O(1)$
TreeMap	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$
Set				
HashSet	$O(1)$	$O(1)$	$O(1)$	$O(1)$
LinkedHashSet	$O(1)$	$O(1)$	$O(1)$	$O(1)$
TreeSet	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$

Source