HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG, DEPARTMENT INFORMATIK

Effizienzvergleich drei verschiedener Implementationen des abstrakten Datentyps SET

Stefan Subotin, Paul Mathia , Dennis Sentler
Philip Scheer

October 18, 2017

Getestet wurden drei Implentationen des abstrakten Datentyps Set. Im folgenden betrachten wir die Implentationen ArrayList, DoubleLinkedSet und HeapList, die über eine Schnittstelle Set, auf Effizienz sowie Vor- und Nachteile getestet wurden. Durch Verifikations- und Quantitative Tests werden diese drei Implementationen gegeneinander verglichen und auf Effizienz analysiert.

1 DOKUMENTATION DER TESTS

In diesem Abschnitt beschäftigen wir uns mit den Verifikationstests die bekräftigen sollen, das die Spezifikation aller drei Implementationen erfüllt ist.

1.1 TESTESIZE

Der erste zu testende Teil war die Implementation der size() Methode. Die size() Methode liefert den aktuellen Zähler über die bereits enthaltenen Elemente. Im Test wurde also überprüft, ob das Hinzufügen sowie entfernen von Datenelementen den eigentlich beabsichtigten Effekt auf den Element-Zähler 'size' hatte. Getestet wurde das Hinzufügen von 10 Elementen in ein anfänglich mit 10 initialisiertes Array. Die Menge wurde anschließend mit 1000 Elementen befüllt und es konnten keine Laufzeitspezifischen Fehler beobachtet werden.

1.2 testeAddAndFind

Im folgenden werden die Methoden add() und find() auf Erfüllung der Spezifikation getestet. Die Methode add() fügt der Menge ein neues Element hinzu und liefert nach Erfolgreicher Speicherung das POS-Objekt mit entsprechendem Index des hinzugefügten Elements. Die find() Methode erwartet als Parameter ein Objekt vom Typ Key, der die Eigenschaft besitzt Elemente in der Menge eindeutig anhand ihres Schlüssels(KEY) zu identifizieren. Getestet wurde erst einmal ob Elemente die der Menge tatsächlich hinzugefügt wurden, auch gefunden werden. In sämtlichen positiven Tests wurde die korrekte POS der hinzugefügten Element zurückgeliefert. Bei den negativ Tests wurde mit Hilfe eines Stopelements überprüft ob das gesuchte Element auch tatsächlich nicht Teil der Menge ist.

1.3 testeAddAndDeletePos

In diesem Teilabschnitt wurden weitere Tests auf die bereits eingeführte Methode add() in Kombination mit deletePos() durchgeführt.Die Methode deletePos() verlangt als Parameter ein Pos-Objekt und entfernt das Element an dem vom Dienstleister übergebenen POS Parameter. Getestet wurden sowohl valide Positionen als auch Positionen die gar nicht im Bereich der Listen Kapazität lagen.

1.4 testeAddAndDeleteKey

Hier erfolgten weitere Tests zum Entfernen von Elementen diesmal allerdings mit der Methode deleteKey, welche als Parameter ein zu einem Element eindeutigen Schlüssel erwartet, und falls vorhanden dieses entfernt. Getestet wurden auch hier valide sowie invalide Positionen getestet und es konnten keine Fehler über JUnit Tests beobachtet werden.

1.5 testeAddAndRetrieve

Die Methode retrieve() liefert - angenommen die im Parameter übergebene Pos ist gültig - das Element an der jeweiligen Pos. Die Methode retrieve() arbeitet in Ihren Grundzügen ähnlich wie die anfänglich beschriebene Methode find(). Auch in diesem Test auf Spezifikation wurden alle zuvor hinzugefügten Elemente wiedergefunden. Der Negativfall lief genauso unproblematisch und ohne spezielle oder unerwartete Vorkommnisse ab.

1.6 testeUnify

Die Vereinigungsmenge einer (nicht notwendigerweise nichtleeren) Menge U von Mengen ist die Menge der Objekte, die in mindestens einem Element von U enthalten sind. Die mathematische Richtigkeit wurde in unserem Fall eingehalten, sodass zwei Mengen die über die unify() Methode vereinigt wurden nach ihrer Vereinigung keine Duplikate enthielten und die size()Methode den richtigen Wert der neu erzeugten Menge lieferte.

2 Auswertung Quantitative Tests