

1 ArrayList

1.1 Einleitung

Ein **ArrayList** implementiert die Liste Interface. ArrayList unterstützt dynamische Arrays, die ihrer Größe sich ändert wenn nötig.

Normale Arrays haben eine bestimmte Größe, wenn sie erzeugt sind dann dieser Größe kann nicht verändert werden. Die haben die Nachteil, dass die Anzahl der Elementen sollen schon bekannt sein.

ArrayLists löst dieser Problem, bei die automatische Größe Veränderung von den Array. Wenn ein ArrayList schon voll ist, und ein neues Element eingefügt ist, wird die ArrayList automatisch die Größe erhöhen, sodass es Platz für die neues Element gibt. Wenn Elementen gelöscht sind, es ist möglich der Array Größe zu verkleinern.

In ArrayList gibt es drei Konstruktoren:

- ArrayList() Erzeugt eine leere array Liste
- ArrayList(Collection c) Erzeugt eine array Liste mit den Elementen von dem Collection c
- ArrayList(int capacity) Erzeugt eine array Liste mit gewünschte Kapazität. Diese Kapazität wurde automatisch größer wenn die array Liste die Kapazität erreicht.

1.2 Methodenbeschreibung

- add(Object o) : - addiert ein Element zu die Liste.
- add(int index, Object o) : - addiert der Element in den eingegebene Stelle.
- addAll(Collection c) : - addiert alle Elementen in einer Collection.
- addAll(int index, Collection c) : - addiert alle Elementen ab die eingegebene Stelle.
- clear() : - löscht alle Elementen in der Liste.
- contains(Object o) : - gibt boolean Wert ob die gefragte Element in der Liste ist.
- get(int index) : - gibt den Element zurück an der eingegebene Stelle.
- remove(int index) : - Löscht der Element in der eingegebene Stelle.
- set(int index, Object o) : - Ersetzt der Element in der eingegebene Stelle mit einem neuen Element

- `size()` : - gibt die gröÙe der Liste zurück
- `Object[] toArray()` : - erzeugt ein Array mit die Elementen der Liste

1.3 Effizienzanalyse

1.3.1 ExtractMin

Die Methode **extractMin** ist eine Methode wo der minimale Wert wird gesucht, dann gelöscht und am Ende zurückgegeben. Dieser Methode hat einen linearen Aufwand bei allen drei Graphen : $\theta(n)$. Ab dem wert 8.000 ist es zu sehen dass die Worst und Average Case sich stark unterscheiden mit dem Best Case. Dies kann sein weil jeder Element muss durchgesucht werden, verglichen und gelöscht.

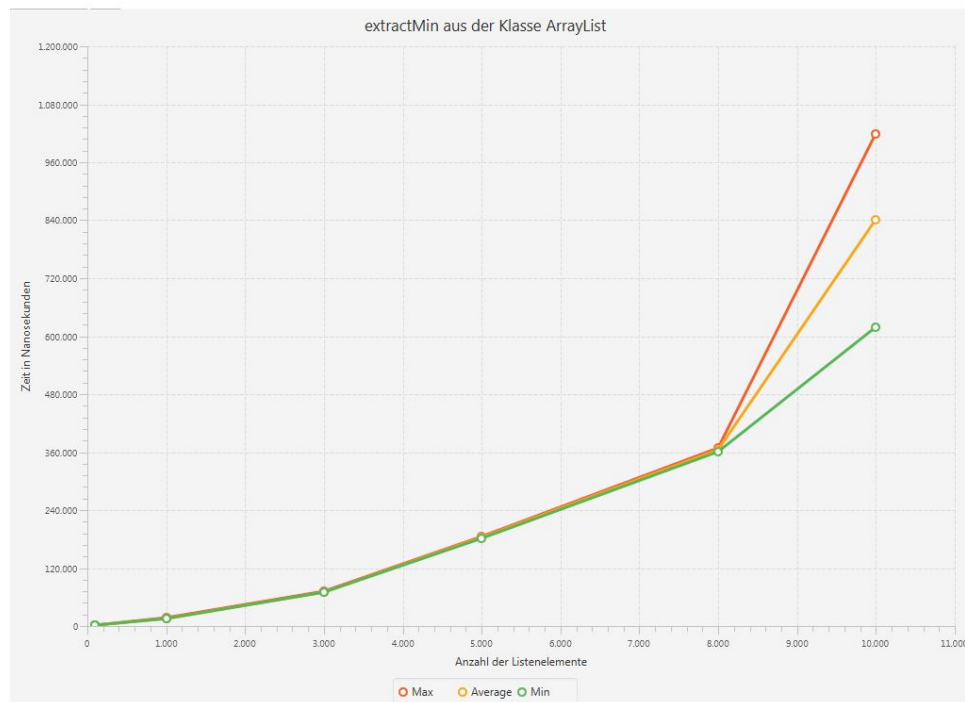


Abbildung 1: ExtractMin

1.3.2 GetMin

GetMin ist eine Methode, in der der niedrigste Wert gesucht wird. Wie bei der letzten Methode **getMin** hat auch ein linearer Aufwand, wie alle drei Graphen zeigen: $\theta(n)$. Bei den drei Fällen ab 5.000 Elementen kann man sehen, dass es einen Unterschied gibt, allerdings nicht so groß.

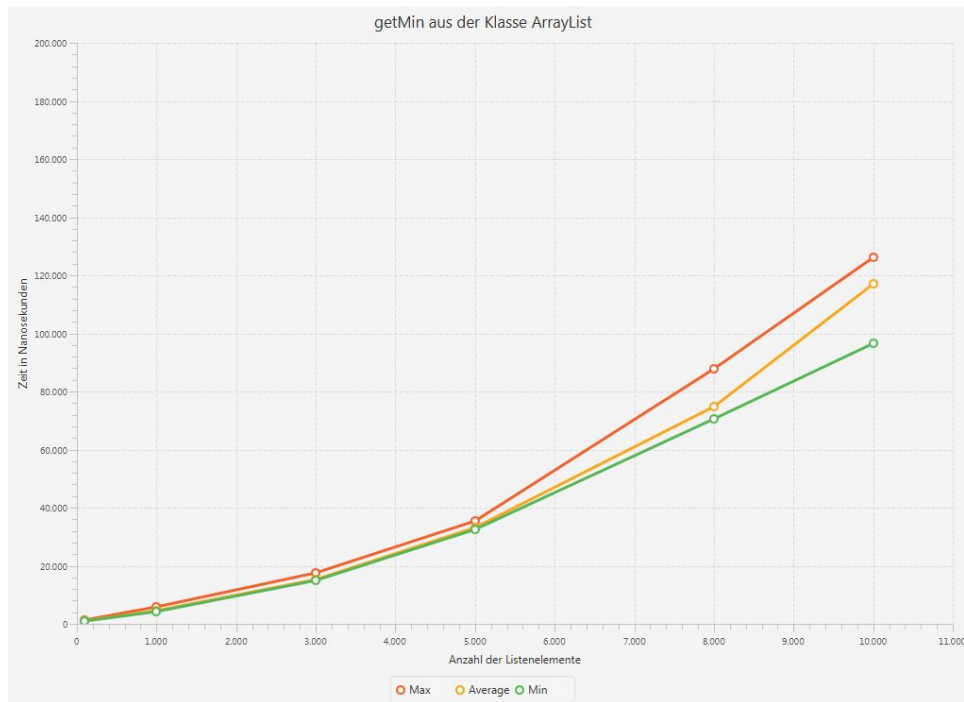


Abbildung 2: getMin

1.3.3 Insert

Insert ist eine Methode, wobei neue Werten zu den ArrayList addiert werden. Wir können sehen, dass die Werte bei allen Graphen fast konstant bleiben. Deswegen hat die Methode **Insert** aus der Klasse ArrayList einen Konstanten Aufwand : $\theta(1)$. Die Insert Methode ist nicht besonderes aufwendig, denn da addiert nur ein wert zu den ArrayList. Bei der Worst Case, ist es zu vermuten dass es hat länger gedauert weil der ArrayList erst vergrößern werden muss um ein neue Wert addieren zu können.

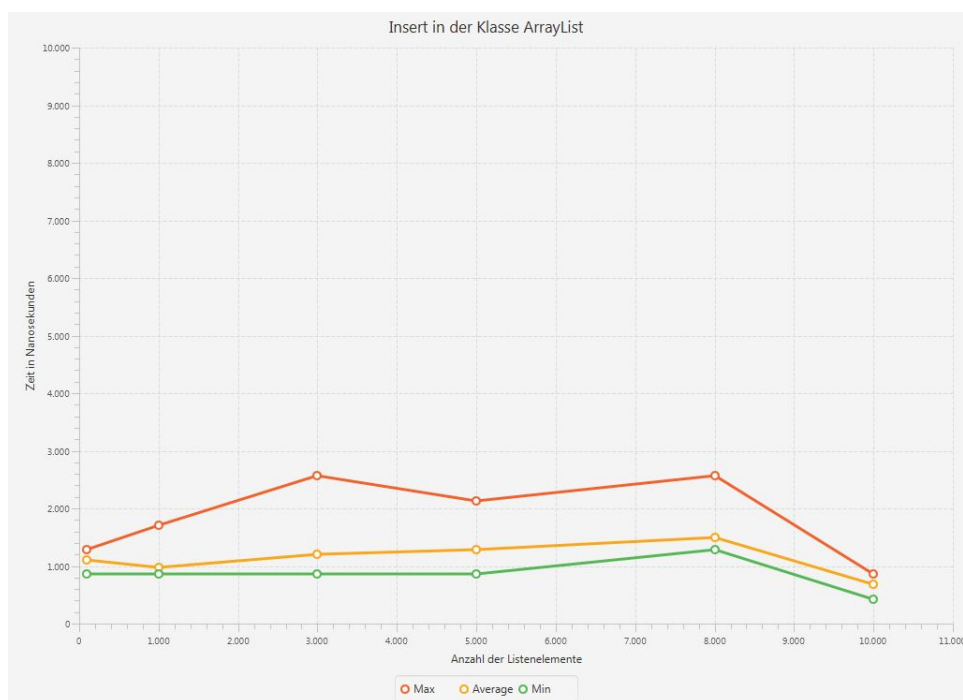


Abbildung 3: Insert

1.3.4 Merge

Merge ist eine Methode wobei ein ArrayList wird zu ein andere ArrayList verbinden. Die Methode **Merge** hat einen Konstanten Aufwand, wie der Grafik zeigt : $\theta(1)$. Wie bei der Insert Methode, der ArrayList wird sich mit ein anderen verbinden, und deshalb ist die Aufwand konstant. Bei dem Worstcase, trifft genau wie bei dem Insert, den ArrayList muss vergrößern um neue werte zu addieren.

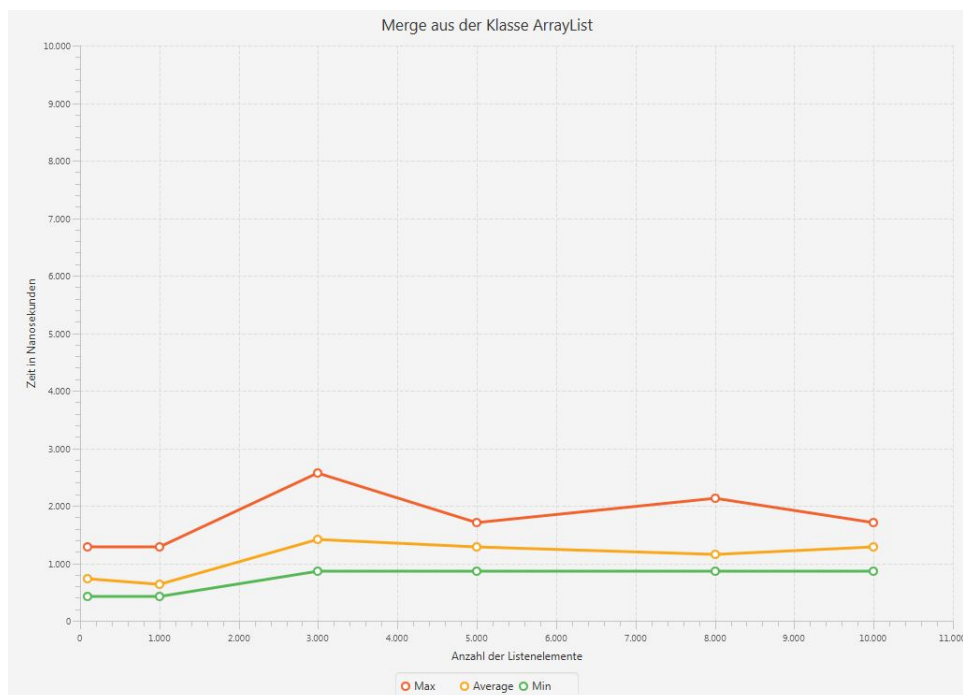


Abbildung 4: Merge

1.3.5 Remove

Remove ist eine Methode, die ein wert von der ArrayList gelöscht wird. Die Methode **Remove** aus der Klasse ArrayList hat einen linearen Aufwand, wie der Grafik zeigt : $\theta(n)$. Die Aufwand kommt dazu da weil die Element die gelöscht werden soll, muss erst gefunden werden, und dann gelöscht. Die Liste wird dann aber weiter suchen für ein andere Treffer.

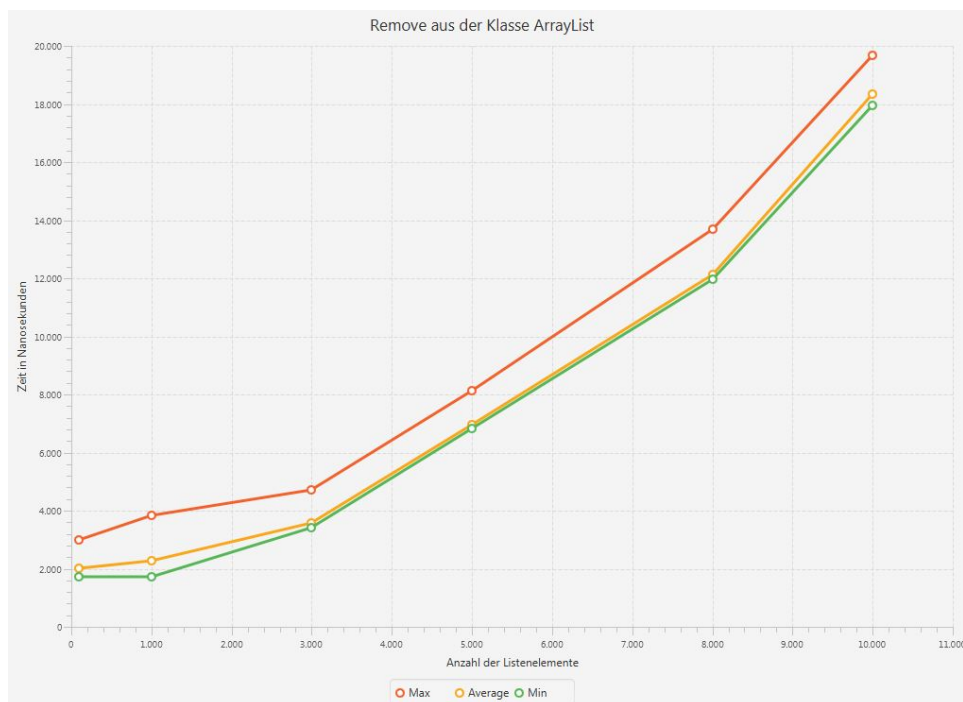


Abbildung 5: Remove

1.4 Schlussfolgerung

Die ArrayList ist wie eine Liste, wobei jeder Element sein eigenes stelle hat. ArrayList ist eine sehr effektive Klasse, die ermöglicht ein Array zu haben, wobei die größer dynamisch ändern kann. Dieser Klasse ist gut zu nehmen, wenn man nicht weiß wie groß der Array sein soll. Im Vergleich mit dem Vector Klasse, ArrayList kann durch mehrere Threads abgerufen werden und bearbeitet.