

1 Queue

1.1 Einleitung

Eine Queue ist eine Warteschlange. In ihr werden Daten in einer bestimmten Reihenfolge abgelegt. Ähnlich wie der Stack braucht auch die Queue nicht viele Methoden, um ihre volle Funktionalität zu erhalten.

Mit Hilfe der Methode *add()* kann ein neues Element der Queue hinzugefügt werden. Die Queue arbeitet nach dem FIFO-Prinzip. FIFO steht für „First In First Out“ und bedeutet, dass das zuerst eingefügte Element auch zuerst wieder ausgegeben wird. Das heißt, neu eingefügte Elemente werden an das Ende der Warteschlange gesetzt.

Um das vorderste Element aus der Queue auszulesen wird die Methode *peek()* benötigt. Diese liest das Element lediglich, wobei die Methode *remove()* das Element nicht nur liest, sondern es außerdem aus der Queue entfernt.

Wie bei jeder Datenstruktur ist es auch bei der Queue sinnvoll, eine Methode *size()* zu implementieren, die die Länge der Warteschlange zurückliefert. In Java ist die Queue ein Interface, das sich auf die Super-Interfaces *Collection* und *Iterable* zurückführen lässt.

1.2 Methodenbeschreibung

In diesem Abschnitt werden die wichtigsten Methoden der Queue beschrieben, die von Java zur Verfügung gestellt werden.

- *add(E item)*
 - fügt das als Argument übergebene Element der Queue an letzter Stelle hinzu
- *peek()*
 - liefert das vorderste Element der Queue zurück ohne es aus der Warteschlange zu löschen
- *remove()*
 - liefert das vorderste Element der Queue zurück und löscht es aus der Warteschlange
- *size()*
 - gibt die Anzahl der Elemente der Queue zurück

1.3 Effizienzanalyse

Bei der Methode „extractMin“, zeigen alle drei Graphen einen linearen Aufwand. Sie steigen auch alle gleichmäßig an. Daran kann man klar herauslesen, dass es sich hier um einen linearen Aufwand handelt.

Linearer Aufwand $\mathcal{O}(n)$

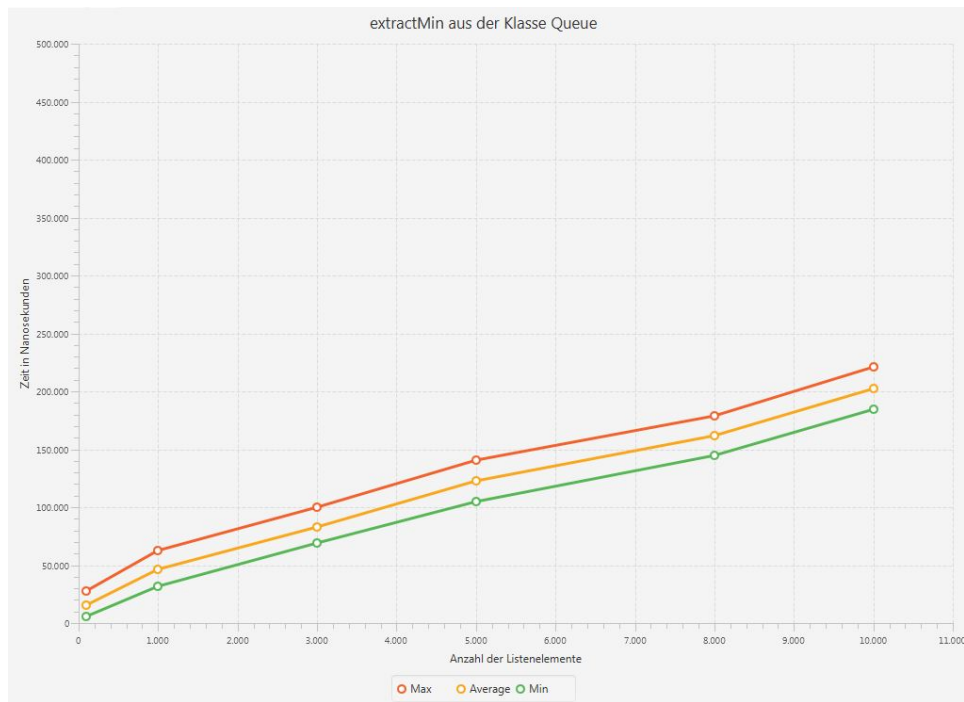


Abbildung 1: ExtractMin

Bei der Methode „getMin“ sieht man bei dem Graphen für die Maximalwerte, dass zwei Punkte (bei 3000 und 5000 Listenelementen) ausreißen. Die Anfangs- und auch die Endwerte dieses Graphen sind sehr linear gehalten, genauso wie die Graphen für die Minimal- und Durchschnittswerte. Daher lässt sich auch hier feststellen, dass der Aufwand linear ist. Linearer Aufwand $\mathcal{O}(n)$

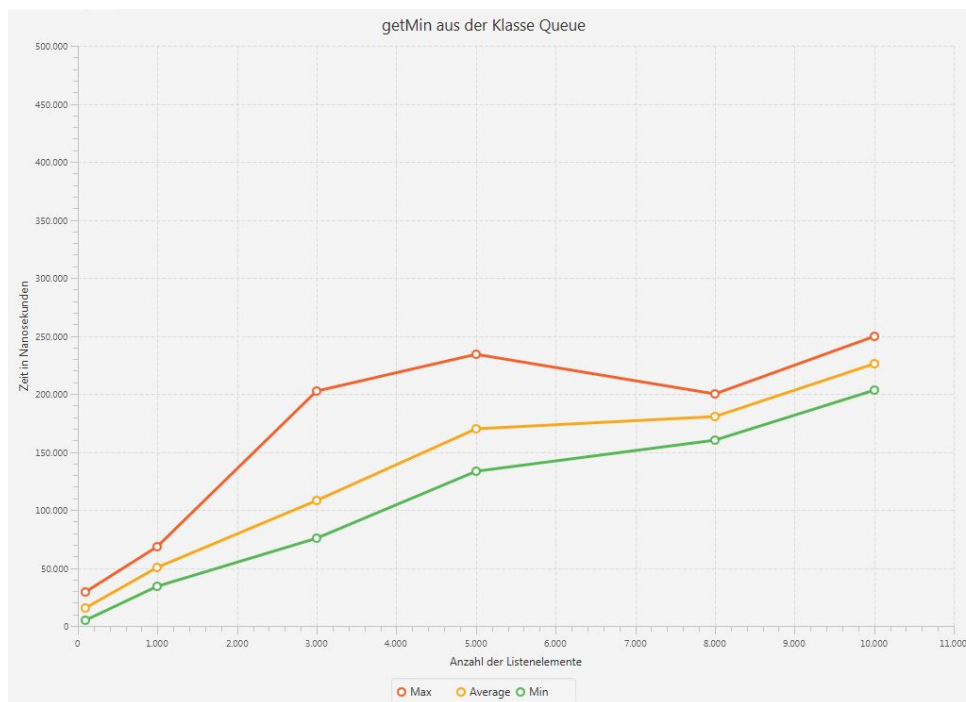


Abbildung 2: getMin

Bei der Methode „insert“ ist der Durchschnittsgraph recht konstant gehalten. Hier gibt es nur beim ersten und zweiten Wert einen leichten Anstieg, den man aber vernachlässigen kann. Beim grünen Graphen sieht man sehr deutlich, dass alle Werte exakt auf einer Linie sind und den gleichen Wert besitzen. Bei dem Graphen für die Maximalwerte liegt die zeitliche Differenz nur 500 Nanosekunden auseinander, daher kann man sagen, dass diese Methode einen konstanten Aufwand besitzt.
Konstanter Aufwand $\mathcal{O}(1)$

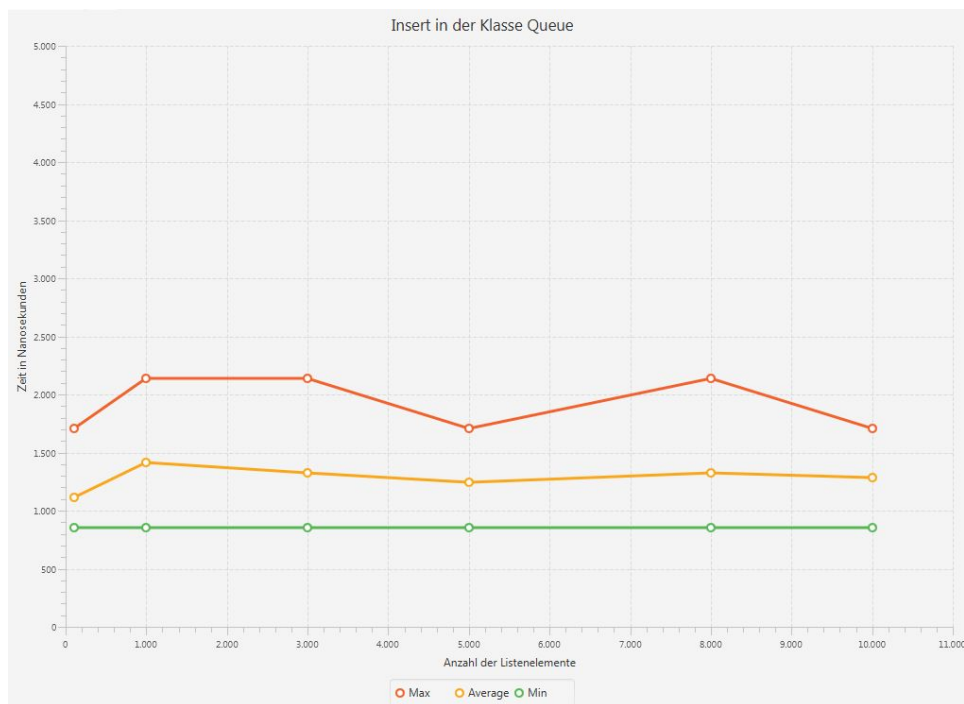


Abbildung 3: Insert

Die Methode „merge“ weist bei dem grünen Graphen eine leichte Steigung auf. Der mittlere Graph sieht recht linear aus, zudem könnte man eine Linie ziehen und die Punkte die über, bzw unter dieser Linie sind, haben den gleichen Abstand zu dieser Linie. Der Graph für die Maximalwerte schwankt stark und zeigt zum Ende einen großen Anstieg. Zieht man alle Graphen in Betracht kommt man auch hier zum Ergebnis, dass es sich um einen linearen Aufwand handelt, auch wenn die Ergebnisse mehr schwanken. Linearer Aufwand $\mathcal{O}(n)$

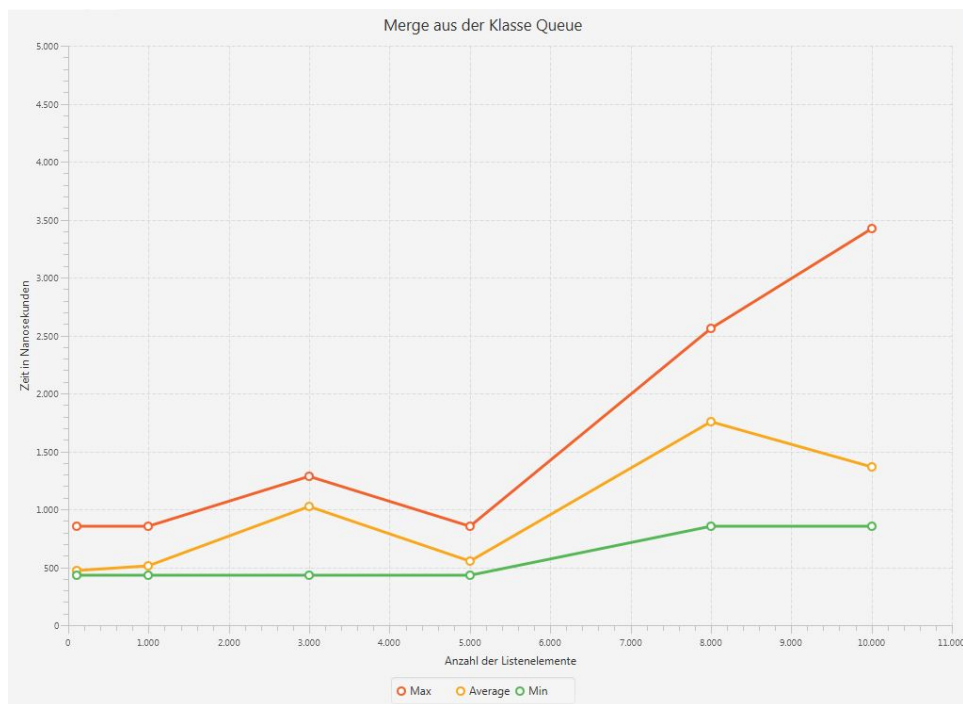


Abbildung 4: Merge

Die Methode „remove“ hat bei dem Graph für die Minimalwerte an der stelle 5000 Listenelemente seinen Maximalwert erreicht und bei mehr Listenelementen sinkt der Graph leicht. Auch bei der Durchschnittskurve liegt das Maximum bei 5000 Listenelementen. Bei letzten Graph sieht man deutlich einen sehr starken anstieg bis 5000 Listenelementen, danach bleibt die Zeit die benötigt wird konstant. Somit kann man hier sagen, dass es sich um einen logarithmischen Aufwand handelt.

Logarithmischer Aufwand $\mathcal{O}(\log n)$

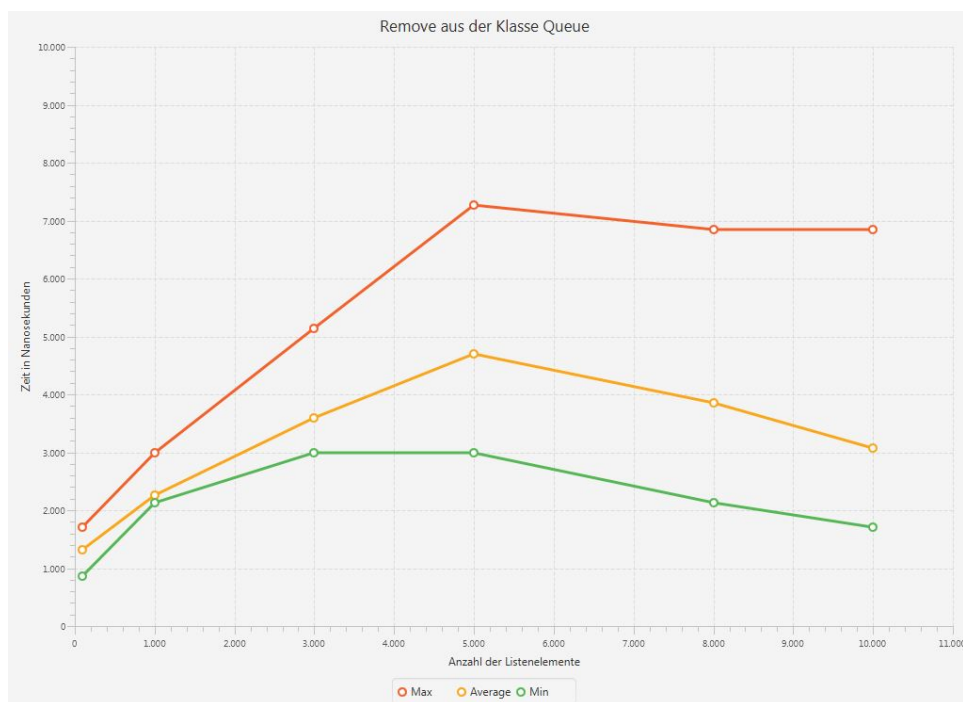


Abbildung 5: Remove

1.4 Schlussfolgerung

Insgesamt ist die Queue von ihrem Aufwand her sehr gut, da der Aufwand meist linear ist. Die Methode „insert“ ist hier besonders zu empfehlen, da der Aufwand konstant ist. Der logarithmische Aufwand der „remove“ Methode ist für wenige Listenelemente nicht so gut, sobald es aber um große Mengen geht kann man auch diese gut verwenden.

Die einzige Methode die gar nicht für die Queue geeignet ist, ist „decrease-Key“, da es nicht möglich war diese zu Implementieren.