

Zu allen Aufgaben liegen am **Pult** Zettel mit **Hilfestellungen**, sofern du bei einer Aufgabe nicht weiter weißt.

- ① Überlege, welche Anordnung der Zahlen 1,2,3,4,5,6,7 besonders viele Operationen benötigt, um mit dem Quicksort-Algorithmus sortiert zu werden.
 - Schreibe die Zahlenfolge auf.
 - Male das Rekursionsdiagramm auf.
 - Schreibe auf jeder Ebene, wie viele Listen-Operationen ausgeführt werden.
 - Führe alle oberen Aktionen erneut aus für eine Anordnung, die besonders wenige Operationen benötigt.
- ② Erweitere das Programm aus der letzten Stunde so, dass du für den Quicksort- und Insertionsort-Algorithmus bei steigender Zufallszahlenanzahl die Anzahl der notwendigen Listenoperatoren angezeigt bekommst. Dein Ergebnis sollte dabei ähnlich aussehen wie die Tabelle unten.
Gehe dabei wie folgt vor:
 - Erweitere die Listenklasse so, dass sie mitzählt, wie oft die Methoden *next()*, *toFirst()*, *toLast()*, *getContent()*, *setContent()*, *insert()*, *append()*, *concat()*, *remove()* aufgerufen werden. Diese Zahl soll durch eine neue Methode *getOperation()* abgefragt werden können.



Hilfestellungen

Ihr erhaltet eine neue *Sortieren.java*, welche zusätzlich die Berechnung mittels Insertionsort enthält. Sofern ihr Aufgabenteil 1 nicht schafft, ist zusätzlich die erweiterte *List.java* verfügbar. Für Aufgabenteil 2 müsst ihr *Listentest.java* erweitern, wofür eine unfertige Vorlage mit

Zahlen	Operationen Quicksort	Operationen Insertionsort
1000	4006	506954
2000	8000	1989557
...

- ③ Vergleiche die Ergebnisse mit der asymptotischen theoretischen Laufzeit.
 - Erstelle in Excel eine Tabelle mit 5 Spalten wie unten.
 - Trage die von dir berechneten Anzahl an Operationen ein.
 - Berechne mit Excel die theoretische Anzahl Operationen.
 - Plote 2. und 3. Spalte sowie 3. und 4. jeweils in einem gemeinsamen XY-Diagramm (erste Spalte als X-Werte).
 - Interpretiere das Ergebnis.

Zahlen	Quicksort	$n^2/2$	Insertionsort	$n \cdot \log n$
1000	1000	506954	4006	3000
2000