technische universität
dortmund

Fakultät für Informatik
Lehrstühle 2, 11

Prof. Dr. C. Sohler, A. Krivošija
H. Blom, N. Kriege, H. Timm, B. Zey
http://tiny.cc/dap2p

Sommersemester 2014

DAP2 Praktikum – Blatt 2

Ausgabe: 11. April — Abgabe: 21.–25. April

Studienleistung (Scheinkriterien)

- Zum Bestehen des Praktikums muss jeder Teilnehmer die folgenden Leistungen erbringen:
  - Es müssen mindestens 50 Prozent der Punkte in den Kurzaufgaben erreicht werden.
  - Es müssen mindestens 50 Prozent der Punkte in den Langaufgaben erreicht werden.
  - Man muss an mindestens 12 Terminen anwesend sein und aktiv mitarbeiten (Nach Absprache mit den Tutoren ist es auch möglich, Fehltermine in anderen Gruppen nachzuholen, sofern dort Platz ist).
- An Feiertagen suchen Sie sich bitte einen Ersatztermin, ansonsten wird dies als „nicht erschienen“ gewertet.

Hinweis zu den Langaufgaben

Für diese Aufgaben stehen Ihnen zwei Wochen Bearbeitungszeit zur Verfügung. In den Praktika dieser Woche können (und sollen) Sie mit der Bearbeitung beginnen. Den Rest müssen Sie außerhalb des Praktikums erledigen. Neben den Lernräumen können Sie dafür auch die Poolräume (z.B. in einer Freistunde) nutzen, wenn dort gerade keine Veranstaltungen stattfinden. Im Praktikum in zwei Wochen präsentieren Sie Ihrem Tutor dann Ihre Lösungen auf Ihrem Praktikumsrechner im Pool und bekommen Punkte dafür (nach Absprache mit dem Tutor ist es auch möglich, die Lösung auf Ihrem eigenen Rechner zu präsentieren). **Wichtig:** Sie müssen die Aufgaben **vor** Beginn des Praktikums (in dem Sie die Lösungen präsentieren) fertig haben. `Sie dürfen die Langaufgaben in Gruppen von bis zu drei Studierenden bearbeiten und präsentieren.`

Ebenfalls wichtig: Der Quellcode ist natürlich mit sinnvollen Kommentaren und die Schleifen mit zugehörigen Invarianten (inkl. Bereich in dem die Invariante gilt) zu kommentieren.

Langaufgabe 2.1 (4 Punkte)

Lernziel: Zufallszahlen und Laufzeitmessung

Diese Aufgabe besteht aus mehreren Teilen. Bitte lesen Sie sich die Aufgabe vorher komplett durch und überprüfen Sie nach der Bearbeitung, ob Sie nichts vergessen haben.

- Legen Sie eine Klasse `Sortierung` an.
- Schreiben Sie eine Methode `public static void insertionSort(int[] array)`, die den Algorithmus InsertionSort aus der Vorlesung implementiert.

- Schreiben Sie eine Methode `public static boolean isSorted(int[] array)`, welche überprüft, ob das Array `array` aufsteigend sortiert ist.
- Stellen Sie mittels `Assertions` die Korrektheit Ihres Programms, insbesondere der Invarianten sicher (siehe Hinweise: Assertions).
- Schreiben Sie die `main`-Methode, die den Algorithmus wie folgt testet.
  - Das Programm bekommt als erstes Argument die Feldgröße übergeben und als optionales zweites Argument die Befüllungsart dieses Feldes:
    `java Sortierung 10000 rand`
    Beachten und behandeln Sie auch Fehleingaben bei den Übergabeparametern!
  - Das Feld kann auf-, absteigend oder zufällig mit `int`-Werten gefüllt werden. Entsprechend lauten die Werte für das zweite Argument: `auf`, `ab` bzw. `rand`. Wenn kein zweiter Parameter übergeben wurde, soll das Array mit zufälligen Werten gefüllt werden. Beachten Sie hierzu die Hinweise zu den *Zufallszahlen*.
  - Ergänzen Sie die `main`-Methode, sodass die Methode `insertionSort` das befüllte Array sortiert. Dabei soll die Zeit gemessen werden, die das Programm braucht, um das Array vollständig zu sortieren. Beachten Sie hier die Hinweise zur *Messung von Laufzeiten*.
  - Anschließend soll das Array auf korrekte Sortiertheit getestet und entsprechend `"Feld sortiert!"` bzw. `"Feld NICHT sortiert!"` ausgegeben werden.
  - Außerdem soll die benötigte Zeit und bei höchstens 100 Elementen auch das Feld ausgegeben werden.

Langaufgabe 2.2 **(4 Punkte)**

Lernziel: Teile & Herrsche mit MergeSort

Implementieren Sie in `public static void mergeSort(int[] array)` den rekursiven MergeSort Algorithmus aus der Vorlesung. Sie können natürlich eigene Hilfsfunktionen schreiben.

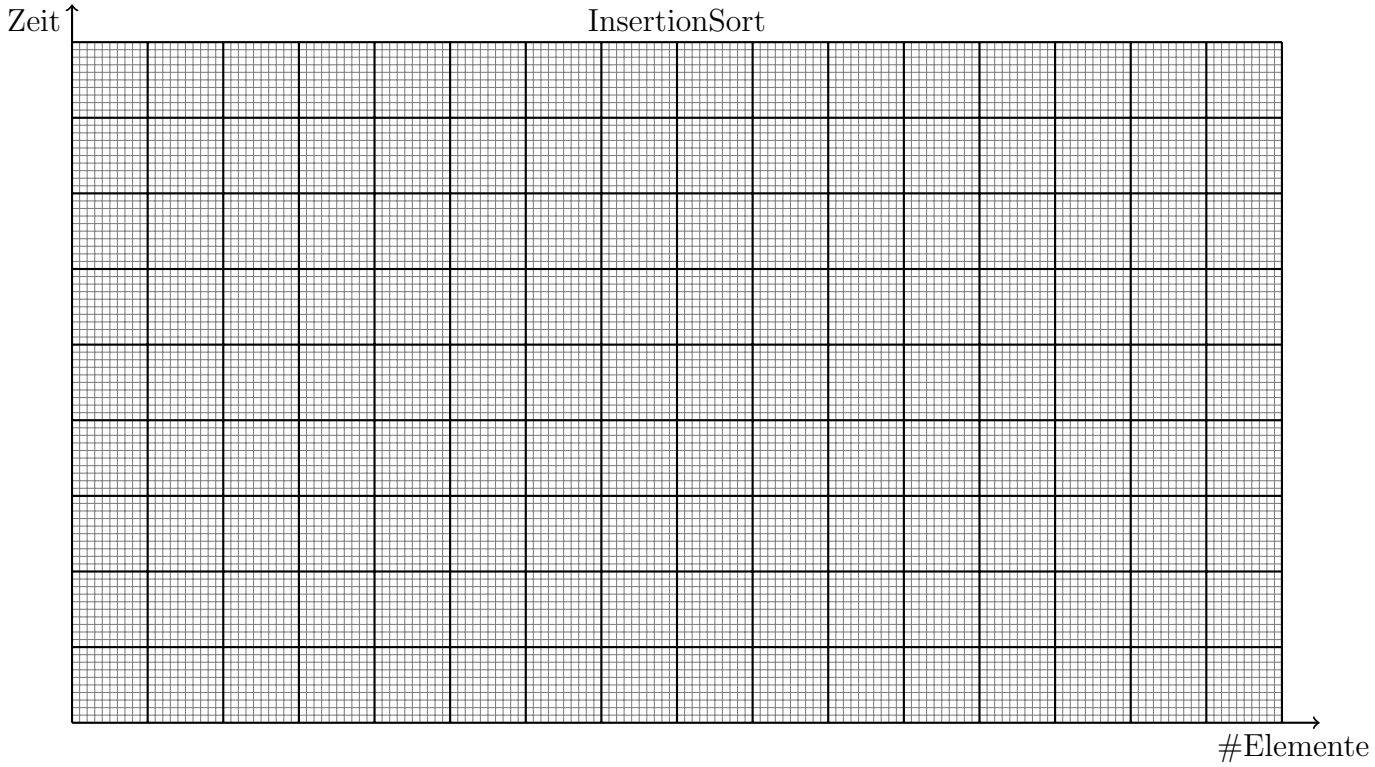
Erweitern Sie die `main`-Methode, sodass jetzt ein zweiter Übergabeparameter angibt, ob InsertionSort oder MergeSort zum Sortieren benutzt werden soll. Die möglichen Parameter sollen `insert` bzw. `merge` sein. Wurde kein Parameter übergeben, soll die Sortierung mit MergeSort auf einem mit Zufallszahlen befülltem Feld erfolgen. Die Wahl der Befüllung des Arrays soll jetzt als dritter Parameter zur Verfügung stehen.

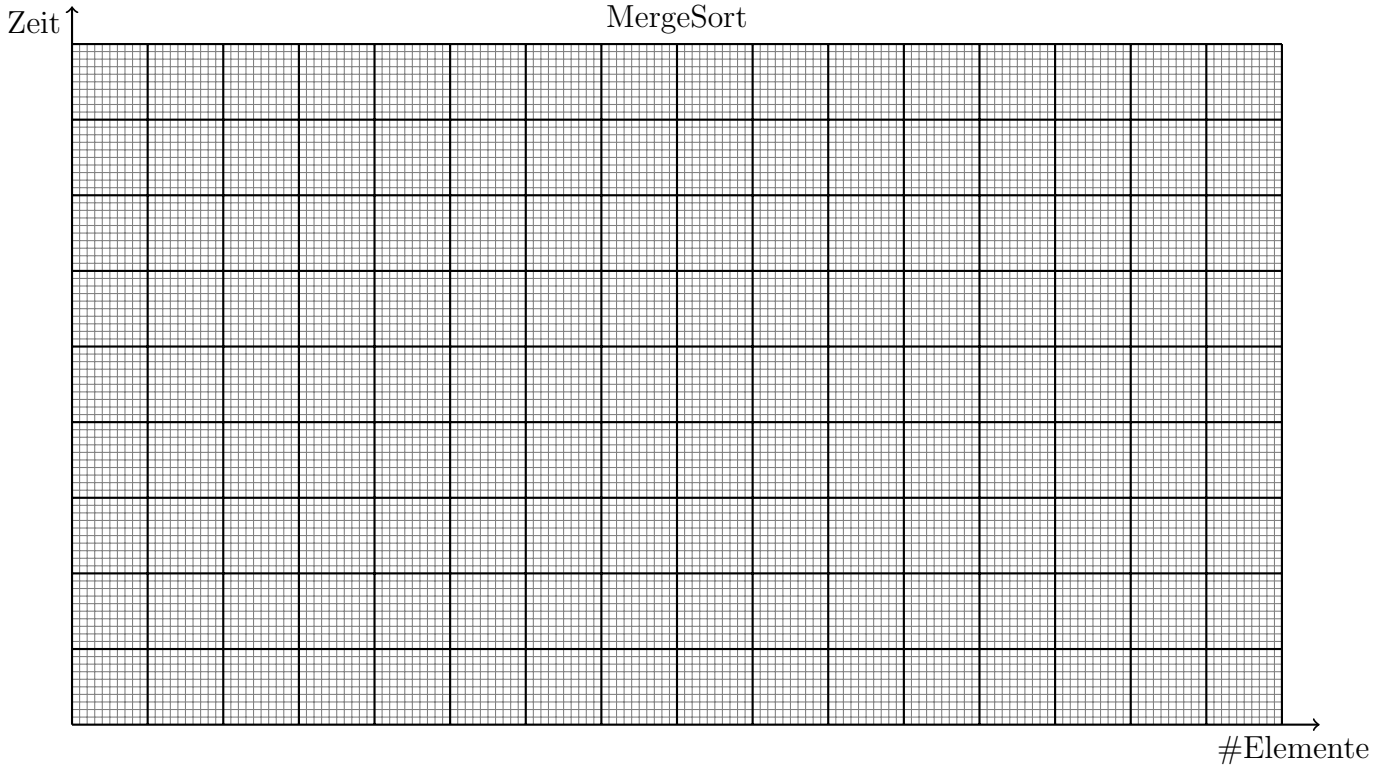
```
java Sortierung 10000 [insert|merge [auf|ab|rand]]
java Sortierung 10000 insert ab
```

Die Laufzeitmessung, die Überprüfung der Parameter, der Test auf Sortiertheit und die Ausgaben haben wie in Aufgabe 2.1 zu erfolgen.

Stellen Sie auch bei dieser Aufgabe mittels `Assertions` die Korrektheit Ihres Programms, insbesondere der Invarianten sicher (siehe Hinweise: Assertions).

Betrachten Sie InsertionSort und MergeSort im Vergleich. Überlegen Sie sich sinnvolle Testeingaben (insb. die Länge der selbigen), und begründen Sie, warum Sie diese für Sinnvoll halten, um die Laufzeiten von InsertionSort und MergeSort zu vergleichen. Ermitteln Sie mehrmals die Laufzeiten für verschiedene Feldlängen und tragen Sie den Mittelwert für jede Feldlänge (jeweils in sinnvollem Maßstab) in die Raster auf der folgenden Seite ein.





Beachten Sie die Hinweise und Tipps auf den folgenden Seiten.

Hinweise und Tipps

2.1 Messung von Laufzeiten

Um Laufzeiten in Programmen zu messen, kann man die Methode `System.currentTimeMillis()` benutzen. Diese gibt die Systemzeit in Millisekunden zurück und hat den Rückgabetyp `long`. Ein Beispielcode zur Laufzeitmessung sieht dann wie folgt aus:

```
...
long tStart, tEnd, msecs;
...
// Beginn der Messung
tStart = System.currentTimeMillis();

// Hier wird der Code ausgefuehrt, dessen Laufzeit gemessen werden soll

// Ende der Messung
tEnd = System.currentTimeMillis();

// Die vergangene Zeit ist die Differenz von tStart und tEnd
msecs = tEnd - tStart;
...
```

Beachten Sie, dass Sie keinen Einfluss auf den Garbage-Collector haben: Dieser kann bei einzelnen Messungen zu recht großen Laufzeitunterschieden führen. Daher ist es empfehlenswert, jede Messung mehrmals zu wiederholen und den Mittelwert der Laufzeiten zu bestimmen. Zudem sollten `Assertions` während einer Laufzeitmessung ausgeschaltet sein, da diese das Ergebnis signifikant verfälschen können.

2.2 Zufallszahlen

In Java steht ein Pseudozufallszahlengenerator zur Verfügung. Die Klasse `java.util.Random` stellt den Konstruktor für einen Pseudozufallszahlengenerator, sowie die Methode `nextInt()`, zur Verfügung.

Um Zufallszahlen zu erzeugen, kann wie folgt vorgegangen werden:

```
...
// Den Generator erzeugen (als Seedwert wird die Systemzeit verwendet)
java.util.Random numberGenerator = new java.util.Random();
...
// Wann immer man eine Zufallszahl braucht
int randomNumber = numberGenerator.nextInt();
...
```

Siehe auch:

`http://docs.oracle.com/javase/6/docs/api/java/util/Random.html`

2.3 Assertions

In Java gibt es das Konstrukt `Assertions`. Diese „Zusicherungen“ oder auch „Annahmen“ helfen dabei, Fehler im Programm zu finden, indem man z.B. Invarianten direkt im Code (in der Form von `Assertions`) mit angibt und diese dann zur Laufzeit überprüft werden.

Syntax :

```
assert exp1 ;
```

oder:

```
assert exp1 : exp2 ;
```

Wobei `exp1` immer ein boolscher Ausdruck sein muss - ist dieser `true`, so ist die `Assertion` korrekt, also die „Annahme“ wahr. Falls `exp1` nicht erfüllt wird, so kann mit `exp2` eine Fehlermeldung erzeugt werden, welche auf der Konsole ausgegeben wird.

Beispiel :

```
...
int laenge;
...
assert laenge > 0 : laenge + " ist nicht groesser 0" ;
int[] array = new int[ laenge ];
...
```

Bei laenge = -5 wäre die Fehlermeldung auf der Konsole dann: `-5 ist nicht groesser 0`

Um `Assertions` bei der Ausführung zu aktivieren, muss dem Interpreter der zusätzliche Parameter `-enableassertions` oder `-ea` mitgegeben werden.

Beispiele:

```
java -enableassertions Klasse
java -ea Klasse param1 param2 param3
```

Wichtig: Zur Messung von Laufzeiten sollten `Assertions` deaktiviert sein, da diese die Ergebnisse verfälschen würden.

Weitere Informationen zu `Assertions` finden sich unter :

`http://download.oracle.com/javase/6/docs/technotes/guides/language/assert.html`