# Übung 11: Sortieren

Abgabetermin: 03.06.2014

**Name:** _____ **Matrikelnummer:** _____

**Gruppe:** ☐ **G1** Di 10:15 ~~☐ **G2** Di 11:00~~ ☐ **G3** Di 12:45

| Aufgabe | Punkte | gelöst | abzugeben schriftlich | abzugeben elektronisch | Korr. | Punkte |
|---|---|---|---|---|---|---|
| Aufgabe 1 | 10 | ☐ | Java-Programm | Java-Programm | ☐ | |
| Aufgabe 2 | 10 | ☐ | Java-Programm | Java-Programm | ☐ | |
| Aufgabe 3 | 4 | ☐ | Testergebnisse | | ☐ | |

#### Aufgabe 1: HeapSort

Implementieren Sie das Sortierverfahren HeapSort mit folgender Schnittstelle (alles *public*):

```
class HeapSort extends Sorter {
  Comparable[] sort(Comparable[] items) { ... }
}
abstract class Sorter {
  Comparable[] sort(Comparable[] items);
}
```

Die *sort*-Methode sortiert die *Comparable*-Objekte im übergebenen Array. Da HeapSort ein Verfahren für *in place*-Sortierung ist, soll die Sortierung ausschließlich in diesem Array geschehen und es sollen keine zusätzlichen Arrays angelegt werden. Daher soll auch das übergebene Array zurückgegeben werden.

Verwendungsbeispiel:

```
Sorter heapSort = new HeapSort();
Integer[] values = new Integer[] {  10, 3, 1, 7, 6, 5, 4, 2, 9, 8 };
Comparable[] sorted = heapSort.sort(values);
for (int i = 0; i < sorted.length; i++) {
  Out.print(" " + sorted[i]);
}  // Ausgabe: 1 2 3 4 5 6 7 8 9 10
```

Abzugeben ist: Java-Programm

#### Aufgabe 2: MergeSort

Implementieren Sie das Sortierverfahren MergeSort mit folgender Schnittstelle (alles *public*):

```
class MergeSort extends Sorter {
  Comparable[] sort(Comparable[] items) {  ... }
}
```

Die *sort*-Methode wird verwendet wie in Aufgabe 1. Implementieren Sie zunächst eine Methode *mergeSorted(...)*, die zwei bereits sortierte Arrays in ein sortiertes Array zusammenfügt:

```
Comparable[] mergeSorted(Comparable[] a, Comparable[] b) { ... }
```

Implementieren Sie dann *sort(...)* so, dass es das übergebene Array in zwei Hälften teilt und auf jede Hälfte wiederum rekursiv *sort(...)* aufruft, wo dieses Array erneut halbiert und *sort(...)* auf die Hälften aufgerufen wird, und so weiter, bis das geteilte Array aus höchstens einem Element besteht. Bei der schrittweisen Rückkehr aus den rekursiven Aufrufen sollen die nun sortierten Teilarrays mit *mergeSorted(...)* zusammengefügt und zurückgegeben werden.

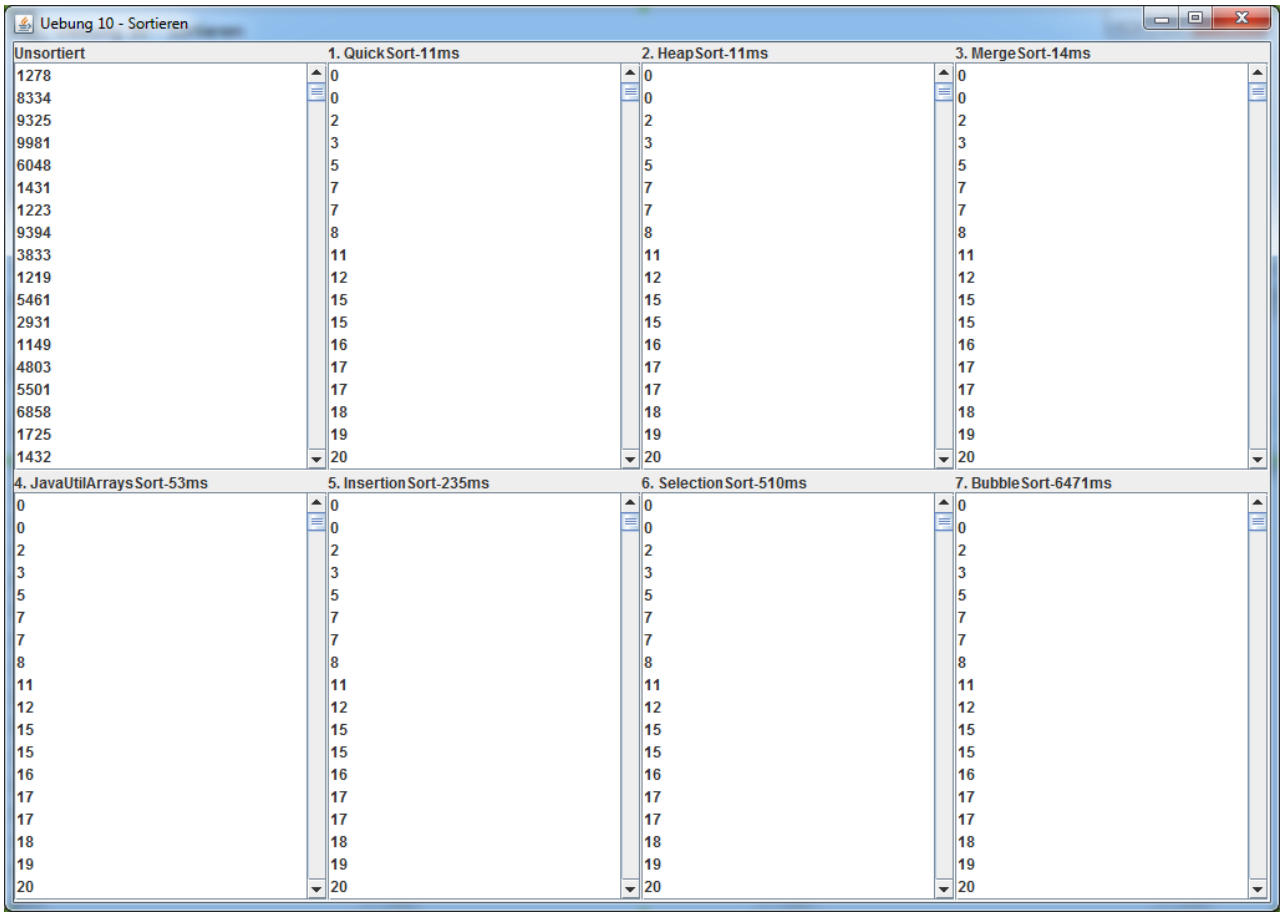
Abzugeben ist: Java-Programm

**Aufgabe 3: Vergleich der Sortieralgorithmen**

Testen Sie Ihre Implementierungen mit dem vorgegebenen Programm *SortTest.java*. Dieses Programm sortiert wiederholt 10.000 Zufallszahlen, bestimmt die benötigte Zeit und vergleicht Ihren *HeapSort* und *MergeSort* mit weiteren Sortierverfahren.

Hinweis: Abhängig von ihrem Computer kann es etwas dauern, bis sich das Ergebnisfenster öffnet.

Beispielausgabe:




Abzugeben ist: Testergebnisse