Übung 07: Sortieren

Sortieren basiert im Wesentlichen auf zwei Operationen: dem Vergleich von zwei Objekten, und, je nach Sortierverfahren, der Tausch von zwei Objekten bzw. das Entfernen und Wiedereinfügen von Objekten. Diese beiden Operationen werden in geeigneter Weise wiederholt, bis die Sortierordnung hergestellt ist. In der Vorlesung haben wir Insertion-, Selection-, Merge- und Quicksort kennen gelernt. In der Übung heute wollen wir Bubble-Sort implementieren, bei dem einzelne Elemente "nach oben blubbern," bis sie an der richtigen Stelle sind. Die folgende Animation veranschaulicht das Verfahren: https://commons.wikimedia.org/w/index.php?curid=14953478

Aufgabe 1: Swap: Elemente tauschen

- Erstellen Sie eine Klasse `Sortieren`
- Schreiben Sie eine statische generische Methode `swap`, welche ein Array sowie zwei Indizes entgegen nimmt, und die Elemente an eben diesen Indizes vertauscht.
- Implementieren Sie einen Testfall (z.B. in einer Klasse `SortierenTests`), der die Funktion der `swap` Methode verifiziert. Verwenden Sie dazu die Annotation `@Test` aus JUnit 5.

Aufgabe 2: Sortieren: Bubble-Sort

Implementieren Sie Bubble-Sort, ein Sortierverfahren welches ein Array entgegen nimmt und dieses in-place sortiert, also das übergebene Array modifiziert.

- Vergewissern Sie sich, dass Sie den Algorithmus verstanden haben, indem Sie die Zahlenfolge 5, 2, 3, 6 "von Hand" mit diesem Verfahren sortieren, vgl https://de.wikipedia.org/wiki/Bubblesort
- Implementieren Sie Bubble-Sort als generische statische Methode, welche ein Array entgegen nimmt.
- Setzen Sie die Bounds so, dass nur Arrays von Klassen übergeben werden können, welche `Comparable` entsprechend implementieren.
- Modifizieren Sie die Klasse `Student` so, dass Objekte dieses Typs der Matrikelnummer aufsteigend nach sortiert werden.
- Schreiben Sie einen Testfall, welcher vier Studenten sortiert. Hinweis: Sie können z.B. die `Arrays.toString(...)` Methode verwenden, um Arrays einfach auf der Konsole auszugeben, sowie `Assertions.assertArrayEquals(...)` um zwei Arrays auf Inhaltsgleichheit zu testen.

Aufgabe 3: Eigene Sortierordnungen

Oft ist es nötig, Arrays (oder andere Datenstrukturen) nach verschiedenen Kriterien zu sortieren, z.B. beim Email-Posteingang: nach Absender, Datum, Betreff, Größe, etc.

- Implementieren Sie Bubble-Sort nun erneut, nur dieses mal ohne Bounds; erweitern Sie dazu die Signatur um einen Comparator und passen Sie Ihre Implementierung entsprechend an.
- Implementieren Sie einen Comparator, welcher Studenten dem Namen nach alphabetisch aufsteigend sortiert. Testen Sie diesen mit einem geeigneten Testfall.
- Schreiben Sie einen Comparator, welche Studenten zuerst dem Namen nach alphabetisch aufsteigend, und bei Namensgleichheit nach Matrikelnummer aufsteigend sortiert. Testen Sie diesen mit einem geeigneten Testfall.