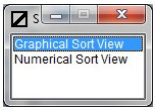
SortDemo Anleitung

1. Einleitung

SortDemo ist ein erweiterbares Java-Programm zur Visualisierung der Abläufe von in-place Sortier-Algorithmen, wie *SelectionSort, InsertionSort, BubbleSort, QuickSort, HeapSort* und anderen. Es kann verwendet werden, um die Abläufe bereits programmierter Algorithmen anschaulich darzustellen, aber auch als Testumgebung, um selbst Algorithmen zu programmieren.

2. Benutzung

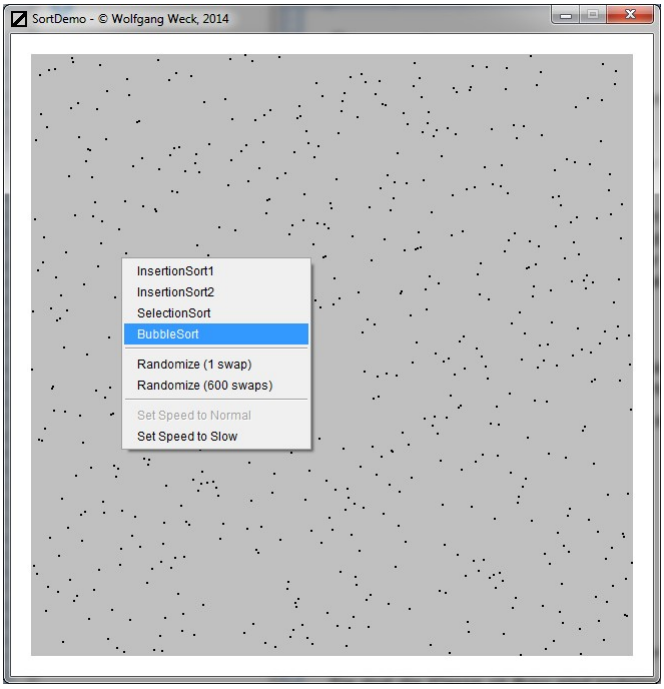


SortDemo wird für die *Algorithmen und Datenstrukturen*-Module als *Gradle*-Projekt ausgeliefert. Nach dem Auspacken und Import in die IDE kann *SortDemo* über den *Gradle*-Task *run* oder direkt über die *main*-Methode der Klasse *Application* gestartet werden.

Stehen mehr als eine Visualisierung zur Verfügung, wird nach dem Programmstart zunächst eine Auswahlliste angezeigt. Nach der Auswahl schliesst sich das Hilfsfenster wieder und die ausgewählte Visualisierung wird geöffnet. Um später eine andere Visualisierung zu benutzen, muss das Programm neu gestartet werden.

Derzeit stehen zwei Visualisierungen zu Auswahl. Die *Graphical Sort View*, mit welcher auf einer zwei-dimensionalen Ebene Punkte sortiert werden, und die *Numerical Sort View*, in der eine kurze selbstdefiniert Zahlenfolge sortiert wird.

2.1. Graphical Sort View



Die graphische Visualisierung zeigt die Werte der zu sortierenden Folge als Punkte. Die Y-Koordinate des Punktes entspricht dabei seinem Wert, die X-Koordinate seiner Position in der Folge. Ist die Folge vollständig sortiert, ergibt sich eine aufsteigende Diagonale von links unten nach rechts oben.

Die *Graphical Sort View* wird über das Kontext-Menu gesteuert, das mit dem rechten Maus-Knopf aktiviert wird.

Im obersten Abschnitt des Menus werden die Namen der zur Verfügung stehenden Algorithmen zur Ausführung angeboten.

Der Abschnitt darunter bietet die Möglichkeit, zufällig ausgewählte Werte bzw. Punkte zu vertauschen, um unsortierte Daten zu erhalten.

Im untersten Abschnitt des Menus kann zwischen zwei Geschwindigkeitsstufen (*normal* und *langsam*) gewählt werden.

Ein laufender Sortieralgorithmus kann jederzeit mit einem Maus-Klick (linke oder rechte Taste) angehalten werden. Mit einem weiteren Maus-Klick wird das Sortieren dann fortgesetzt. Wird im angehaltenen Zustand die rechte Maus-Taste gedrückt, erscheint ein Menu, das die Wahl zwischen Fortsetzung des Algorithmus und definitivem Abbruch anbietet.

2.2. Numerical Sort View

Die *Numerical Sort View* zeigt, wie 12 frei gewählte ganze Zahlen sortiert werden.

Die aktuell angezeigten Zahlen können gespeichert und wieder abgerufen werden, um verschiedene Algorithmen in der gleichen Ausgangslage ausprobieren zu können. Ausserdem kann eine zufällige Ordnung erzeugt oder die aktuelle Folge umgekehrt werden.

Mit einem Slider zuunterst im Fenster kann die Geschwindigkeit der Animation verändert werden. Die höchste Geschwindigkeitsstufe sollte normalerweise in kürzester Zeit zu einem Ende des Algorithmus führen, weshalb auf einen separaten Abbruchmechanismus verzichtet wurde.

Nach dem Start eines Sortiervorgangs, werden in der Zahlenreihe farblich folgende Vorgänge dargestellt:

- gelb: die markierten Zahlen wurden vom Algorithmus verglichen
- grün: die markierten Zahlen wurden vom Algorithmus vertauscht und sind nun in Einklang mit ihrer Ordnung im Array platziert
- rot: die markierten Zahlen wurden vom Algorithmus vertauscht und sind nun entgegen ihrer Ordnung im Array platziert

3. Sortier-Algorithmen Programmieren

Um dem Framework einen Programmieralgorithmus hinzuzufügen, muss eine Klasse programmiert werden, die das Interface *SortAlg* implementiert. Dieses definiert eine einzige Methode *void run (SortData d)*. Für Programmieraufgaben in den Algorithmen und Datenstrukturen-Modulen sind entsprechende leere Klassen bereits vorgegeben und im System registriert.

```
import ch.fhnw.algd.sortdemo.framework.SortAlg;
import ch.fhnw.algd.sortdemo.framework.SortData;

public class MySort implements SortAlg {
  public void run(SortData data) {}
}
```

Das beim *run*-Aufruf übergebenen *SortData*-Objekt abstrahiert die zu sortierenden Daten und bietet einen geschützten Zugriff über drei Methoden:

```
public final class SortData {

  // Anzahl der zu sortierenden Werte, gültige Indices sind [0...size()-
1]
  public int size() { ... }

  // Vergleich der Werte mit dem aktuellen Index i und j: d[i] < d[j]
  public boolean less(int i, int j) { ... }

  // Vertauscht die Werte an den Positionen i und j miteinander
  public void swap(int i, int j) { ... }
}
```

4. Konfiguration

In der *Configuration* im default package wird festgelegt, welche Sortier-Algorithmen und Visualisierungen tatsächlich zur Verfügung stehen.

Möchten Sie beispielsweise einen zusätzlichen Sortier-Algorithmus einfügen, für den noch kein Menu-Eintrag angezeigt wird, müssen Sie in die Methode *Configuration.populateAlgDirectory* eine Anweisung der folgenden Art einfügen:

```
SortAlgDirectory.add("My Sort", "ch.fhnw.algd.sortalgs.MySort");
```

Der erste Parameter der *SortAlgDirectory.add*-Methode gibt den im Menu anzuzeigenden Namen an, der zweite Parameter den vollständigen qualifizierten Klassennamen der Implementation des Algorithmus.

Analog werden in der Methode *Configuration.populateViewDirectory* die zur Verfügung stehenden Visualisierungen durch Aufrufe von *SortViewDirectory.add* konfiguriert.

Wenn Sie über längere Zeit nur eine Visualisierung benutzen und den Auswahldialog beim Programmstart umgehen möchten, können Sie in der Methode *Configuration.populateViewDirectory* die Registrierung aller nicht gewünschten Visualisierungen entfernen. Ist nur noch eine Visualisierung konfiguriert, wird der Auswahldialog nicht angezeigt.