Wo sind wir?

Was wissen wir schon?

- objektorientierte Modellierung
- Syntax von Programmiersprachen
- Referenz-/Wert- -zuweisung/-übergabe
- Grundelemente von JAVA

Was kommt jetzt?

- Sortierung
- Induktionsbeweis
- abstrakte Datentypen
- Performanztests

Und in ferner Zukunft?

- Ereignisbehandlung
- nebenläufige Programmierung
- verteilte Programmierung

Sortierung

Was: Elemente gemäß einer Ordnungsrelation sortieren, so daß das kleinste Element am Anfang, das größte am Ende steht
Es gibt einen sortierten und einen unsortierten Teil. Am Anfang ist der sortierte Teil leer, am Ende ist der unsortierte Teil leer.

Wie: Selektionssortierung

- totale Ordnung der Zahlen
- grundlegende Darstellung der Elemente:Feld
- Reihenfolge durch Index des Feldes

Warum:
beweisen, daß wirklich das kleinste Element selektiert wird und der sortierte Teil nicht mehr verändert werden muß.

Sortierung

Sortierung Allgemein können wir das Sortieren definieren als einen Prozeß, der eine ungeordnete Menge in eine geordnete Menge überführt.

Wir brauchen dazu eine Ordnungsrelation, die uns für zwei Elemente der Menge entscheidet, ob sie den gleichen Rang haben, oder das eine Element einen höheren Rang hat als das andere.

Ordnungsrelation

Zahlen, Buchstaben und Wörter haben eine **totale Ordnung** $>$: jedes Element ist verglichen mit jedem anderen Element entweder größer oder kleiner, aber nicht gleichrangig.

Ordnen wir hingegen Aussagen bezüglich ihres Wahrheitswertes, so erhalten wir alle wahren Aussagen, die untereinander alle gleichrangig sind, und alle falschen Aussagen, die untereinander alle gleichrangig sind. Wahre und falsche Aussagen haben unterschiedlichen Rang. Die Ordnung bezüglich des Wahrheitswertes ist also eine **partielle Ordnung**, bei der mehrere Elemente gleichrangig sind.

Vorgehen

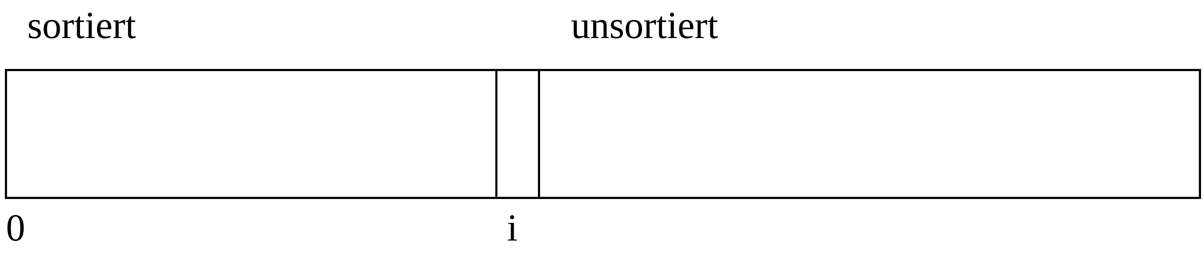
Die Elemente sind in einem Feld gespeichert.

Alle Positionen kleiner i sind sortiert. Wir wollen so vorgehen, daß wir niemals den bereits sortierten Teil wieder bearbeiten müssen.

Im unsortierten Teil (i und aufwärts) das kleinste Element auswählen und an die i-te Position stellen.

Dann sind i Positionen sortiert und wir betrachten nur noch die Positionen $i + 1$ und aufwärts.

Das machen wir, bis es keinen unsortierten Teil mehr gibt.

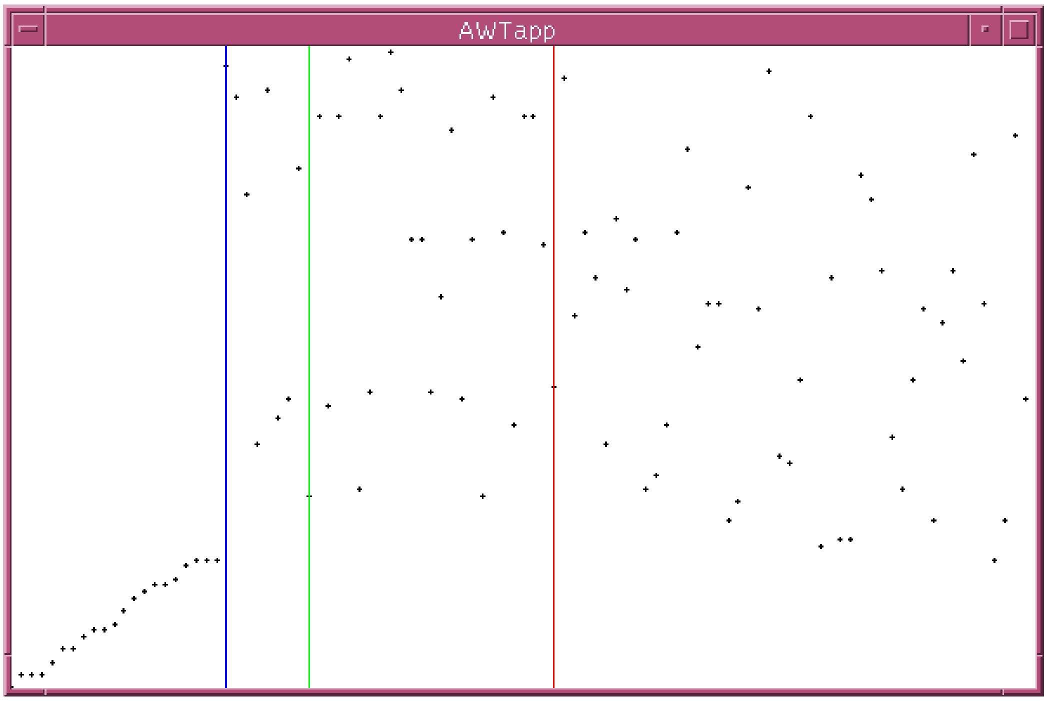


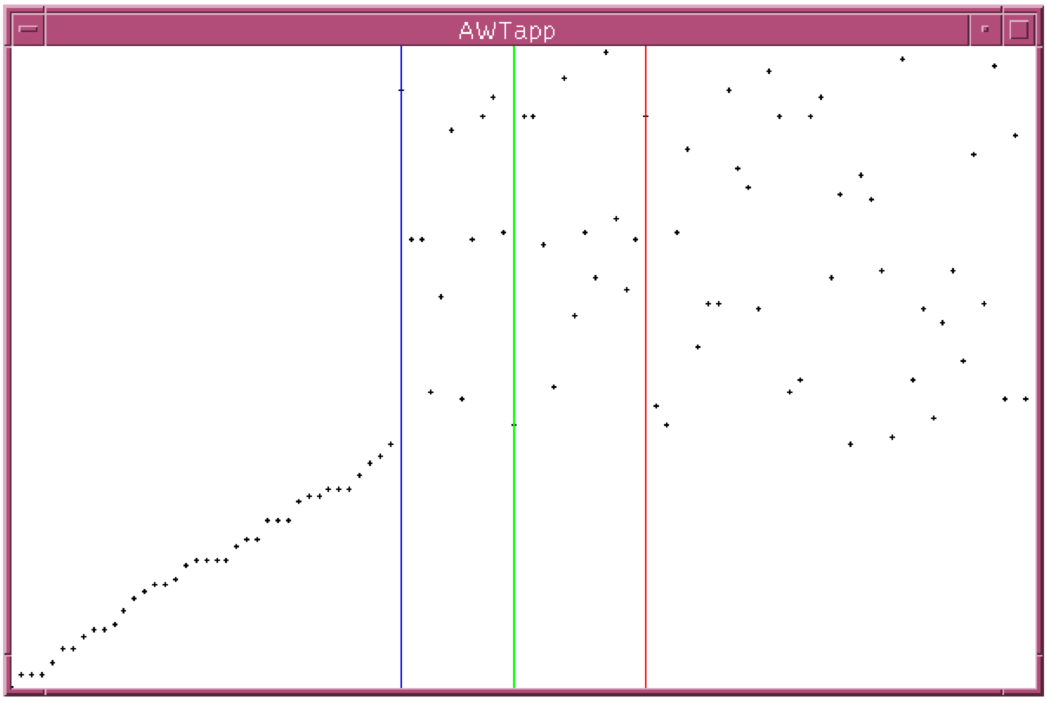
Selektionssortierung

Suche nach dem kleinsten Element im unsortierten Teil:

- Nehmen wir an, das erste Element des unsortierten Teils sei schon das kleinste. $k = i$.
- Mit Zeiger j laufen wir das restliche Feld ab:
  - Ist das nächste Element größer, sind wir bestätigt und nehmen das nächste.
  - Wenn wir ein kleineres Element finden, wird k seine Position und wir sehen noch alle weiteren Elemente an, um festzustellen, ob es ein noch kleineres Element gibt.
- Das kleinste Element – an Position k – wählen wir aus. Wir vertauschen die Positionen von j und k.

i,j,k





Realisierung in JAVA

Wie sollen wir dies Modell der Problemlösung implementieren?

- Welchen Typ sollen die sortierten und unsortierten Teile haben?
- Wie soll die Erweiterung des sortierten gegenüber des unsortierten Teils erfolgen?
- Wie suche ich im unsortierten Teil nach dem kleinsten Element?
- Wie vertausche ich die Elemente?

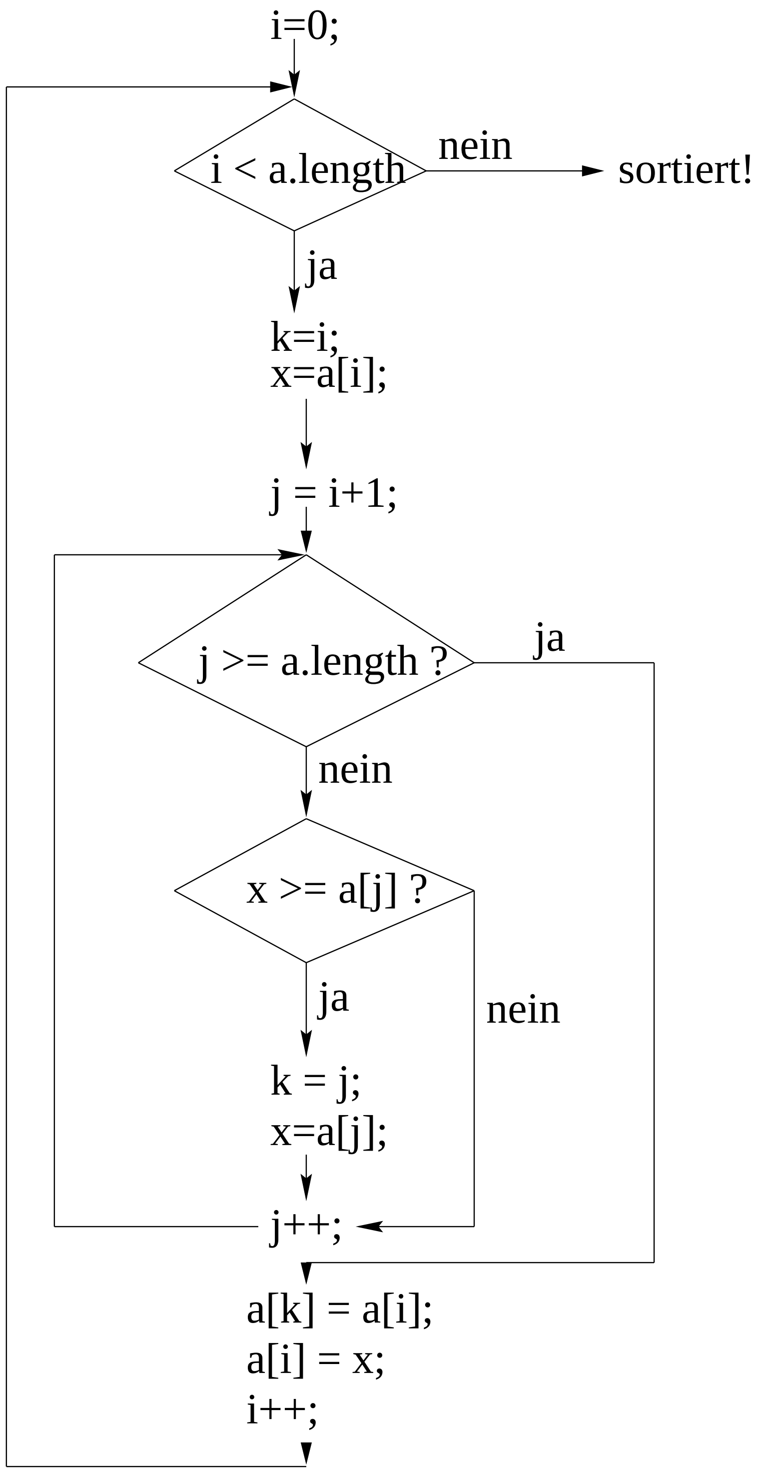
Typ: Feld von Zahlen, $<$ als Ordnungsrelation.

Erweiterung des sortierten Teils: Trennung zwischen sortiert und unsortiert ist i. Erweiterung um l Elemente ist $i = i + l$.

Suche nach kleinstem Element: j läuft durch den unsortierten Teil. k zeigt auf das bisher kleinste Element im unsortierten Teil. Jedes j-Element wird mit dem k-Element verglichen.

Vertauschen: falls j-Element $<$ k-Element, brauchen wir einen Zwischenspeicher und vertauschen die Inhalte der Positionen.

Ablaufdiagramm



```
public class SelectionSort {            // Klasse SelectionSort

    public static void sort (int[] a) {
        for (int i=0; i<a.length-1; i++) {
            int k = i;                  // Index des bisher kleinsten
            int x = a[i];               // Wert des bisher kleinsten
            for (int j=i+1; j<a.length; j++)
            if (a[j] < x) {             // falls kleineres gefunden,
                k = j;                                  // merke Index
                x = a[j];                               // merke Wert
            }
            a[k] = a[i];            // speichere bisher kleinstes um
            a[i] = x;               // neues kleinstes nach vorne
        }
    }
}
```

```
import AlgoTools.IO;

/** testet Sortierverfahren
*/

public class SelectionSortTest {

    public static void main (String argv[]) {

        int[] a;                          // Feld fuer Zahlenfolge

        a = IO.readInts ("Bitte eine Zahlenfolge: ");
        SelectionSort.sort (a);           // SelectionSort aufrufen
        IO.print ("sortiert mit SelectionSort: ");
        for (int i=0; i<a.length; i++) IO.print (" "+a[i]);
        IO.println ();
        IO.println ();
    }
}
```

Ergebnis

`Bitte eine Zahlenfolge: 2 15 10 30 1`

`sortiert mit SelectionSort: 1 2 10 15 30`

bei $i = 0$ sortiert $\emptyset$, unsortiert 2 15 20 30 1

bei $i = 1$ sortiert 1, unsortiert 15 10 30 2

bei $i = 2$ sortiert 1 2, unsortiert 10 30 15

bei $i = 3$ sortiert 1 2 10, unsortiert 30 15

bei $i = 4$ sortiert 1 2 10 15, unsortiert 30