	Sortieren		AnPr	V 1.1
	Name	Klasse	Datum	

1 Grundsätzliches zum Sortieren

Ein Sortierverfahren ordnet Elemente einer Liste oder Arrays nach einer definierten Eigenschaft an. Hierbei kann man diesem Verfahren diverse Eigenschaften zuschreiben:

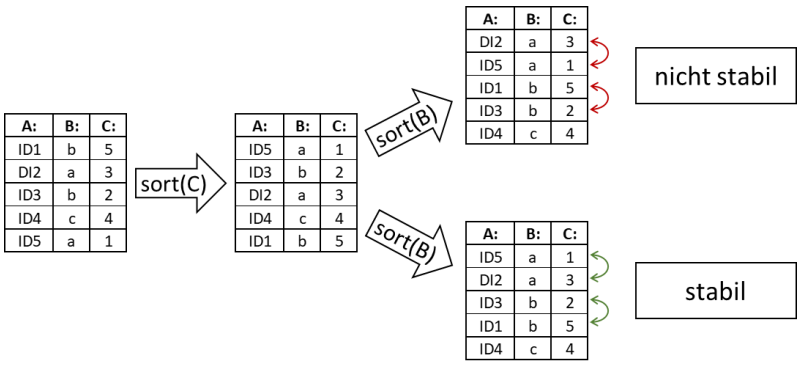
Zeitkomplexität: Wie ändert sich die Sortierzeit bei der Erhöhung von zu sortierenden Elementen?
Platzkomplexität: Wie ändert sich der Platzbedarf bei der Erhöhung von zu sortierenden Elementen?

Wir unterscheiden die durchschnittliche Komplexität, den „best case" (also bei vorsortierten Listen) und den „worst case" (also wenn die Elemente so stehen, dass ein maximaler Aufwand entsteht). Man gibt die die Komplexität oft in der „O-Notation" an. Bspw. Gibt O(n) an, dass die Komplexität sich linear ändert – bspw. der Platzbedarf eines Sortieralgorithmus verdoppelt sich, bei einer Verdoppelung der Elemente.

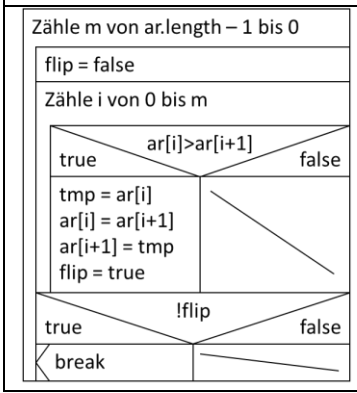
Notation:	**Beschreibung:**	**Beispiel:**
O(1)	Aufwand unabhängig der Arraygröße	Keines bei Sortierungen[1]
O(n)	Linearer Aufwand	Best case Bubblesort
O(n*log n)	Quasi-linearer Aufwand	Quicksort
$O(n^2)$	Quadratischer Aufwand	Worst case Bubblesort

Stabilität: Hier geht es darum, ob bei gleichen Werten des Sortierkriteriums die Reihenfolge der Objekte gleich bleiben. Dies ist vor allem dann notwendig, wenn wir sequentiell nach unterschiedlichen Kriterien sortieren. Im nebenstehenden Beispiel wurde zuerst nach der Spalte C und anschließend nach Spalte B sortiert.

Im nicht stabilen Fall liegen nun in der Spalte C große Werte über kleinen, im stabilen Fall bleibt das Sortierkriterium der Spalte C innerhalb des möglichen Rahmens bestehen.

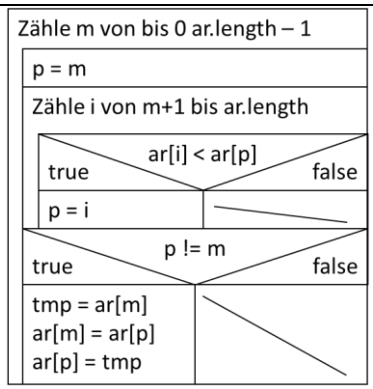


Bubblesort:



Ziel:
Mit dem 1. Durchlauf von m steht der höchste Wert rechts. Beim 2. Durchlauf steht der zweithöchste Wert links daneben usw.

Selection-Sort



Ziel:
Mit dem ersten Durchlauf von m steht der kleinste Wert links. Beim zweiten der zweitkleinste rechts daneben. Getauscht wird nur einmal pro m-Durchlauf.

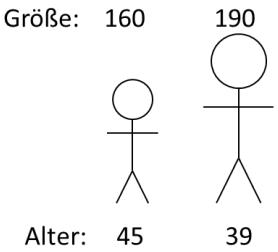
Parallelisierbarkeit: Manche Algorithmen kann man in kleinere, voneinander unabhängige Teilaufgaben zerlegen, welche auf unterschiedlichen Prozessorkernen abgearbeitet werden können.

Iterative/rekursive Umsetzung: Algorithmen, welche rekursive Methoden nutzen sind oft einfacher zu realisieren, benötigen zur Laufzeit aber oftmals mehr Speicherplatz.

[1] O(1) wäre der zeitliche Aufwand beim Anhängen eines Elements an einer verketteten Liste

2 Sortieren von Objekten

Objekte haben oftmals mehr als eine Eigenschaft. Sie können somit nicht ohne weiteres verglichen werden. Gehen wir davon aus, wir wollen Personen sortieren. Hierfür müssen wir erstmal festlegen, wonach sortiert werden soll – nach Größe oder nach Alter.



Um dies nun umsetzen zu können, muss das Objekt eine Methode aufweisen, welche die zu vergleichende Eigenschaft festlegt. Bei dem entsprechenden Vergleich gibt es drei Möglichkeiten:

- Wert A ist größer Wert B
- Wert A ist kleiner Wert B
- Wert A ist gleich Wert B

Eine Implementierung hierzu sieht bspw. für den Altersvergleich wie folgt aus:

```
int compareTo(Person p) {
  if (this.age > p.age) {
    return 1;
  }
  if (this.age < p.age) {
    return -1;
  }
  return 0;
}
```

Die Methode `compareTo` wird in Java durch das Interface `Comparable` gefordert. Implementieren wir dieses Interface (und somit die Methode `compareTo`), so können wir ein Array von Objekten mithilfe der `util` Klasse `Arrays` sortieren:

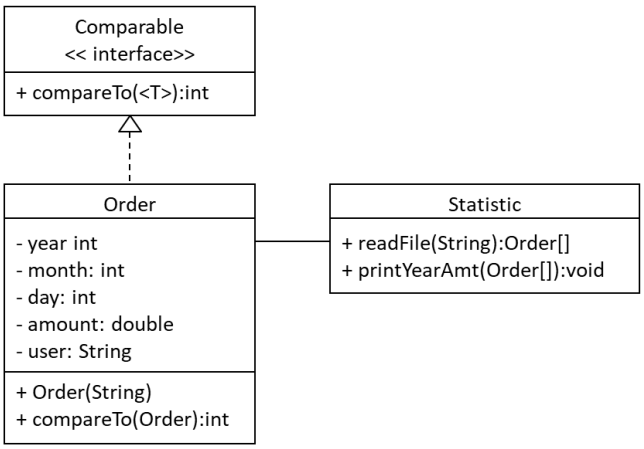
```
Arrays.sort(myPersonArray);
```

3 Umgang mit sortieren Objektlisten

Erstellen Sie nun ein Programm, welches das File `Accounting.csv` liest und es in ein Array von Order Objekten speichert. Danach ergeben Sie die summierten Bestellbeträge pro Jahr aus.

Accounting.csv hat folgendes Format: `YYYY;MM;DD;Amount;USER`

Beispieldaten:
```
2017;4;9;8.21;AIC
2020;7;22;30.73;AIC
2021;3;1;41.29;AIC
2021;9;9;72.21;AIC
```



Die Klasse Order soll einen Konstruktor erhalten, welcher mit einem String einer Zeile des Files `Accounting.csv` aufgerufen wird und daraus die Instanzvariablen belegen kann.

`readFile()` erhält als Parameter den Filepfad zum `Accounting.csv` File, öffnet es und liest es in ein Order Array ein. Nach der Sortierung nach dem Datum gibt es das Array zurück.

`printYearAmt()` gibt auf der Konsole pro Jahr die Summierten `amount` Beträge aus.