Selbststudium OOP10 & ALG5

Aufgaben

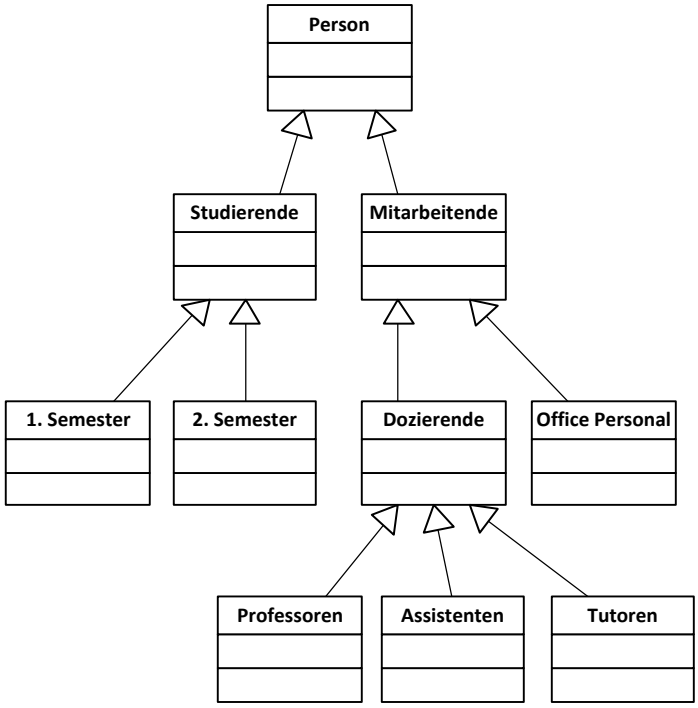
Kapitel 8.1

1. zu bearbeitende Aufgabe: 8.1

done

Kapitel 8.3

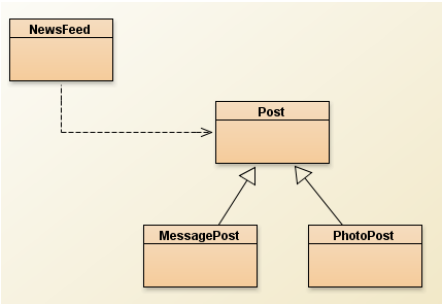
2. zu bearbeitende Aufgabe: 8.3



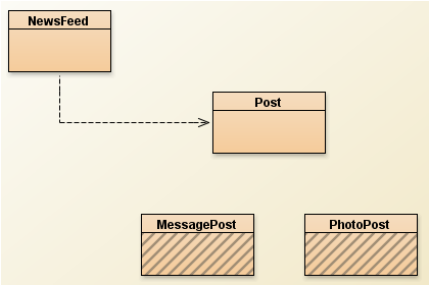
Kapitel 8.4

3. zu bearbeitende Aufgaben: 8.4 und 8.7

8.4:



remove „extends“ ➔



8.7:

Mit dem Befehl „super(author)“ wird der Konstruktor der Basisklasse aufgerufen. Nachdem der Konstruktor der Basisklasse abgearbeitet ist, wird der Konstrukor der eigentlichen Klasse ausgeführt.

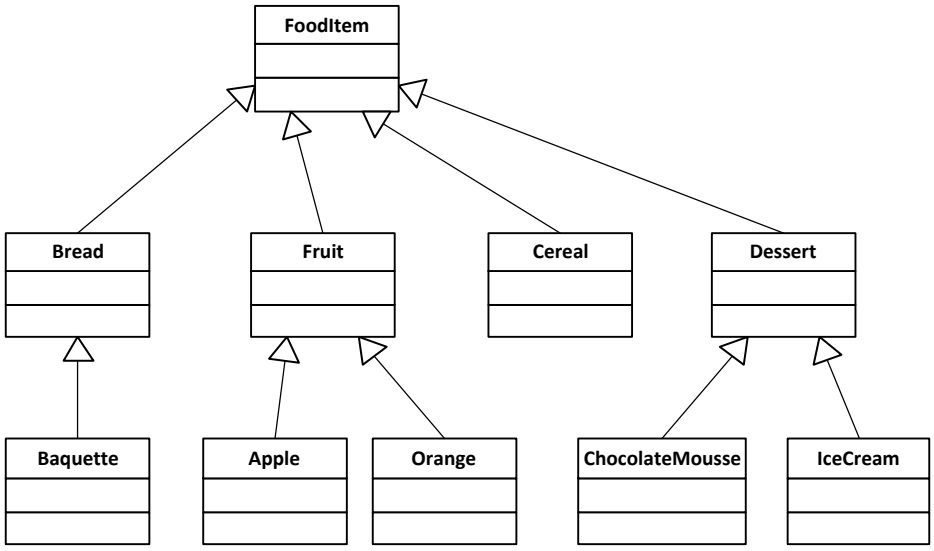
Kapitel 8.5

4. zu bearbeitende Aufgabe: 8.8

```
public class EventPost extends Post
{
   private String date;
   private String description;
   public EventPost(String author, String eventDate, String text)

   {
      super(author);
      date = eventDate;
      description = text;
   }

   public String getDate()
   {
      return date;
   }

   public String getDescription()
   {
      return description;
   }
}
```

Kapitel 8.6

5. zu bearbeitende Aufgabe: 8.9



Kapitel 8.7

6. zu bearbeitende Aufgaben: 8.12 und 8.14

8.12:

a)

Person p1 = new Student();

gültig, Student ist eine Subclass von Person

Person p2 = new PhDStudent();

gültig, PhDStudent ist eine Subclass von Student welche eine Subclass von Person ist

PhDStudent phd1 = new Student();

nicht gültig! Student ist keine Subclass von PhDStudent!

Teacher t1 = new Person();

nicht gültig! Person ist keine Subclass von Teacher!

Student s1 = new PhDStudent();

gültig, PhDStudent ist eine Subclass von Student

b)

s1 = p1;

nicht gültig! Person ist keine Subclass von Student!

s1 = p2;

nicht gültig! Person ist keine Subclass von Student!

p1 = s1;

gültig, Student ist eine Subclass von Person

t1 = s1;

nicht gültig! Student ist keine Subclass von Teacher!

s1 = phd1;

gültig, PhDStudent ist eine Subclass von Student

phd1 = s1;

nicht gültig! Student ist keine Subclass von PhDStudent

8.14:

Es müssen keine Änderungen an der Klasse NewsFeed vorgenommen werden, da NewsFeed ein Post-Objekt erwartet. Und da die Unterklassen (PhotoPost, MessagePost und EventPost) jeweils von der Basisklasse (in diesem Fall Post) abgeleitet sind, wird diese Anforderung erfüllt.

Kapitel 8.11

7. zu bearbeitende Aufgabe: 8.18

gültig:

m = t;

T ist eine Subclass von M

m = x;

X ist eine Subclass von M

o = t;

T ist eine Subclass von O

nicht gültig:

o = m;

M ist KEINE Subclass von O

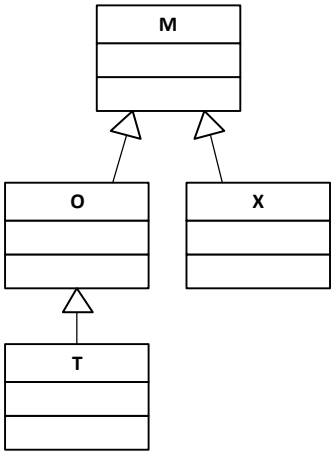
o = x;

X ist KEINE Subclass von O

x = o;

O ist KEINE subclass von X

Somit folgt:



Algorithmen

8. Sortieren Sie auf Papier die nachstehende Folge mit dem klassischen Quicksort Algorithmus (gemäss Implementation in der Klasse Quicksort.java auf ILIAS). Schreiben Sie dabei die Zwischenschritte auf.

8	2	4	9	5	3	5	7	6	10	9	1	13

8	2	4	9	5	3	5	7	6	10	9	1	13

Neues Trennelement: 13 und gleich am endgültigen Platz

8	2	4	9	5	3	5	7	6	10	9	1	13

Neues Trennelement: 1 an endgültige Stelle verschieben

1	8	2	4	9	5	3	5	7	6	10	9	13

Neues Trennelement: 9

9 tauscht mit 6

Trennelement 9 an endgültige Stelle verschieben

1	8	2	4	6	5	3	5	7	9	9	10	13

Neue Trennelemente: 7 und 10

TR7: 8 tauscht mit 5

TR7: an endgültige Stelle verschieben

TR10: ist bereits an endgültiger Stelle

1	5	2	4	6	5	3	7	8	9	9	10	13

Neue Trennelemente: 3, 8 und 9

TR3: 5 tauscht mit 2

TR3: an endgültige Stelle verschieben

TR8: ist bereits an endgültiger Stelle

TR9: ist bereits an endgültiger Stelle

1	2	3	5	4	6	5	7	8	9	9	10	13

Neue Trennelemente: 2 und 5

TR2: ist bereits an endgültiger Stelle

TR5: 5 tauscht mit 4

TR5: an endgültige Stelle verschieben

1	2	3	4	5	5	6	7	8	9	9	10	13

Neue Trennelemente: 4 und 6

TR4: ist bereits an endgültiger Stelle

TR6: ist bereits an endgültiger Stelle (nichts zu tauschen)

1	2	3	4	5	5	6	7	8	9	9	10	13

Neues Trennelement: 5

TR5: ist an endgültiger Stelle

1	2	3	4	5	5	6	7	8	9	9	10	13

9. Überprüfen Sie Ihr Resultat aus 8. indem Sie z.B. den vorgegebenen Algorithmus mit Konsolenausgaben für die Zwischenresultate ergänzen.

```
Parameter: a, 0, 8
124359876
Parameter: a, 0, 4
124356879
Parameter: a, 0, 3
124356879
Parameter: a, 0, 1
123456879
Parameter: a, 6, 8
123456879
Parameter: a, 6, 7
123456879
```

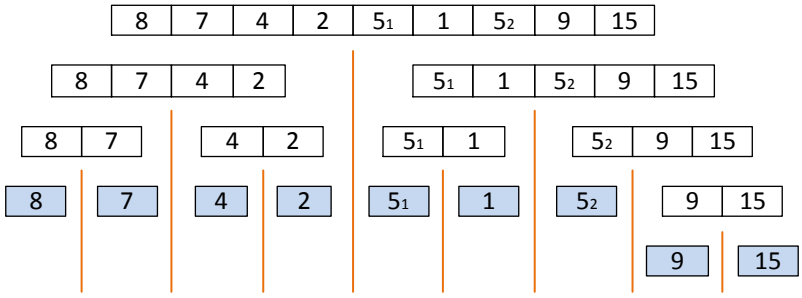
10. Wo würden Sie den Quicksort Algorithmus nicht einsetzen? Definieren Sie zwei Szenarien und begründen Sie Ihre Antwort.

- *Sobald die zu sortierenden Elemente stabil sortiert werden müssen (z.B. Telefonbuch wenn bereits nach Vorname sortiert wurde und nun noch nach Nachname sortiert werden soll)*
- *Wenn O(n ∗ log(n)) nicht überschritten werden darf. → ist die Folge bereits sortiert, benötigt Quicksort $O(n^2)$*

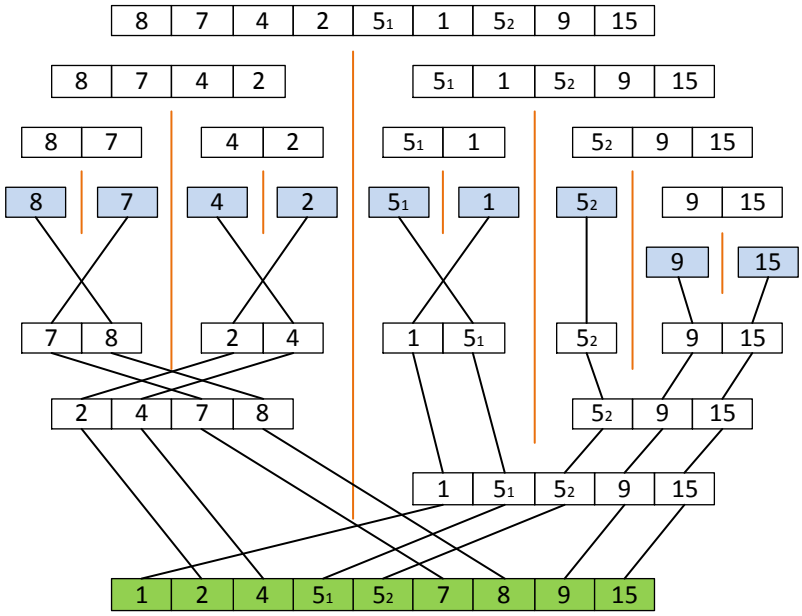
11. Sortieren Sie auf Papier die nachstehende Folge mit dem Mergesort Algorithmus (gemäss der Implementation in Java auf Folie 36). Schreiben Sie dabei die Zwischenschritte auf.

8	7	4	2	5	1	5	9	15

Aufteilen:



Zusammenfügen (jeweils zuerst die Elemente von Links)



OPTIONAL

Wiederholen Sie das Sortieren (Quicksort), verwenden Sie jetzt aber das "median-of-three" Verfahren zur Bestimmung des Trennelementes. Hinweis: Tauschen Sie vor dem Trennen das letzte Element der (Teil-) Folge mit dem so bestimmten Trennelement (falls notwendig).

8	2	4	9_1	5_1	3	5_2	7	6	10	9_2	1	13

median-of-three: 8, 5, 13 => 8 (8 ist das Trennelement / Pivotelement)

8	2	4	9_1	5_1	3	5_2	7	6	10	9_2	1	13

8 mit 13 tauschen

13	2	4	9_1	5_1	3	5_2	7	6	10	9_2	1	8

(Teil-) Folge sortieren, von links beginnen und überprüfen ob das Element grösser als das Trennelement ist, falls ja von rechts das Element suchen welches kleiner ist als das Trennelement und diese zwei miteinander vertauschen.

13	2	4	9_1	5_1	3	5_2	7	6	10	9_2	1	8

13 mit 1 tauschen

1	2	4	9_1	5_1	3	5_2	7	6	10	9_2	13	8

9 mit 6 tauschen

1	2	4	6	5_1	3	5_2	7	9_1	10	9_2	13	8

Das Trennelement wird zwischen die zwei Teilhälften an seine endgültige Position verschoben:

9 und 8 tauschen

1	2	4	6	5_1	3	5_2	7	8	10	9_2	13	9_1

Die Teilfolgen werden aufgeteilt und die neuen Trennelemente berechnet:

median-of-three: 1, 5, 7 => 5

median-of-three: 10, 13, 9 => 10

1	2	4	6	5_1	3	5_2	7	8	10	9_2	13	9_1

Trennelement mit dem letzten Element der Teilfolge tauschen:

5 mit 7 tauschen

10 mit 9 tauschen

1	2	4	6	7	3	5_2	5_1	8	9_1	9_2	13	10

Teilfolge sortieren:

1	2	4	6	7	3	5_2	5_1	8	9_1	9_2	13	10

6 und 3 tauschen

1	2	4	3	7	6	5_2	5_1	8	9_1	9_2	13	10

Trennelement an endgültige Position verschieben:

7 und 5 tauschen

13 und 10 tauschen

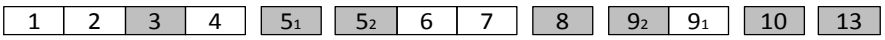
1	2	4	3	5_1	6	5_2	7	8	9_1	9_2	10	13

Teilfolgen auftrennen, Trennelement bestimmen:
median-of-three: 1, 4, 3 => 3
median-of-three: 6, 5, 7 => 5
median-of-three: 9, 9 => 9
13 ist alleine => endgültige Position

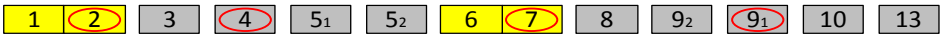
Trennelement mit dem letzten Element der Teilfolge tauschen:
5 und 7 tauschen

Teilfolge sortieren:
keine Elemente zu sortieren

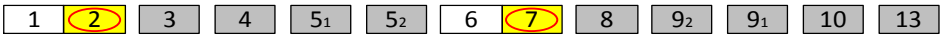
Trennelement an endgültige Position verschieben:
4 und 3 tauschen
6 und 5 tauschen
9 und 9 tauschen



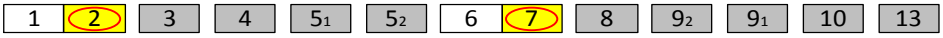
Teilfolgen auftrennen, Trennelement bestimmen:
median-of-three: 1, 2 => 2
4 ist alleine => endgültige Position
median-of-three: 6, 7 => 7
9 ist alleine => endgültige Position



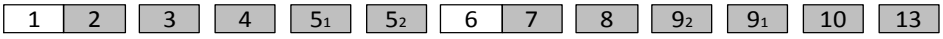
Trennelement mit dem letzten Element der Teilfolge tauschen:
keine Elemente zu tauschen



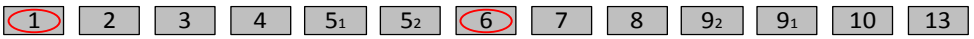
Teilfolge sortieren:
keine Elemente zu sortieren



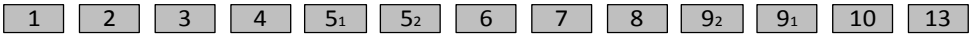
Trennelement an endgültige Position verschieben:
2 ist an endgültiger Position
7 ist an endgültiger Position

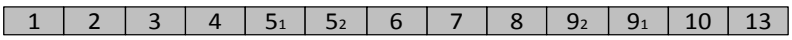


Teilfolgen auftrennen, Trennelement bestimmen:
1 ist alleine => endgültige Position
6 ist alleine => endgültige Position



fertig sortiert:





Wie viele Schritte können Sie so einsparen? Welche Verbesserung bringt das "median-of-three" Verfahren in diesem Beispiel?

Gemäss meinen beiden Beispielen entsteht kein grosser Vorteil des median-of-three Verfahren. Der Vorteil entsteht erst, wenn die Zahlen bereits fast sortiert sind. Da es dann weniger Schritte benötigt.

Normal: 13 Trennelemente
Median-of-three: 13 Trennelemente

Normal: 4 x tauschen
Median-of-three: 4 x tauschen

Normal: 8 Ebenen von Trennelemente
Median-of-three: 5 Ebenen von Trennelemente

Normal: 0 x Vergleich der Trennelemente
Median-of-three: 13 x Vergleich der Trennelemente (+4 x tauschen der Trennelemente)