UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Software Engineering im Wintersemester 2021/2022

Prof. Dr. Martin Leucker, Malte Schmitz, Stefan Benox, Julian Schulz, Benedikt Stepanek, Friederike Weilbeer, Tom Wetterich

Übungszettel 8 (Lösungsvorschlag)

17.12.2021

Abgabe bis Donnerstag, 16. Dezember um 23:59 Uhr online im Moodle.

Aufgabe 8.1: Klassendiagramm und Objektdiagramm

5 Punkte, mittel

In dieser Aufgabe geht es um die Repräsentation des abstrakten Syntaxbaums (AST) für arithmetische Ausdrücke als Java-Datenstruktur und die Ausgabe sowie Auswertung eines ASTs.

Es geht in dieser Aufgabe nur um die Diagramme. *Sie geben also keinen Quellcode ab.* Es kann allerdings durchaus zum Verständnis hilfreich sein, die beschriebenen Datenstrukturen und Methoden tatsächlich umzusetzen.

Folgende Aufrufe sollen funktionieren und jeweils exakt die als Kommentar angegebene Ausgabe produzieren:

```
Expression e = new Division(new Value(100),
        new Addition(new Variable("a"), new Variable("b")));
System.out.println(e); // 100 / (a + b)
System.out.println(e.evaluate(
        Map.of("a", 2, "b", 8))); // 10

Expression sub = new Division(new Variable("a"),
        new Subtraction(new Variable("b"),
                new Addition(new Variable("c"),
                        new Variable("d"))));
Expression add = new Division(new Variable("a"),
        new Addition(new Variable("b"),
                new Addition(new Variable("c"),
                        new Variable("d"))));
```

```
System.out.println(sub); // a / (b - (c + d))
System.out.println(add); // a / (b + c + d)
```

Dazu sollen folgende Strukturen angelegt werden:

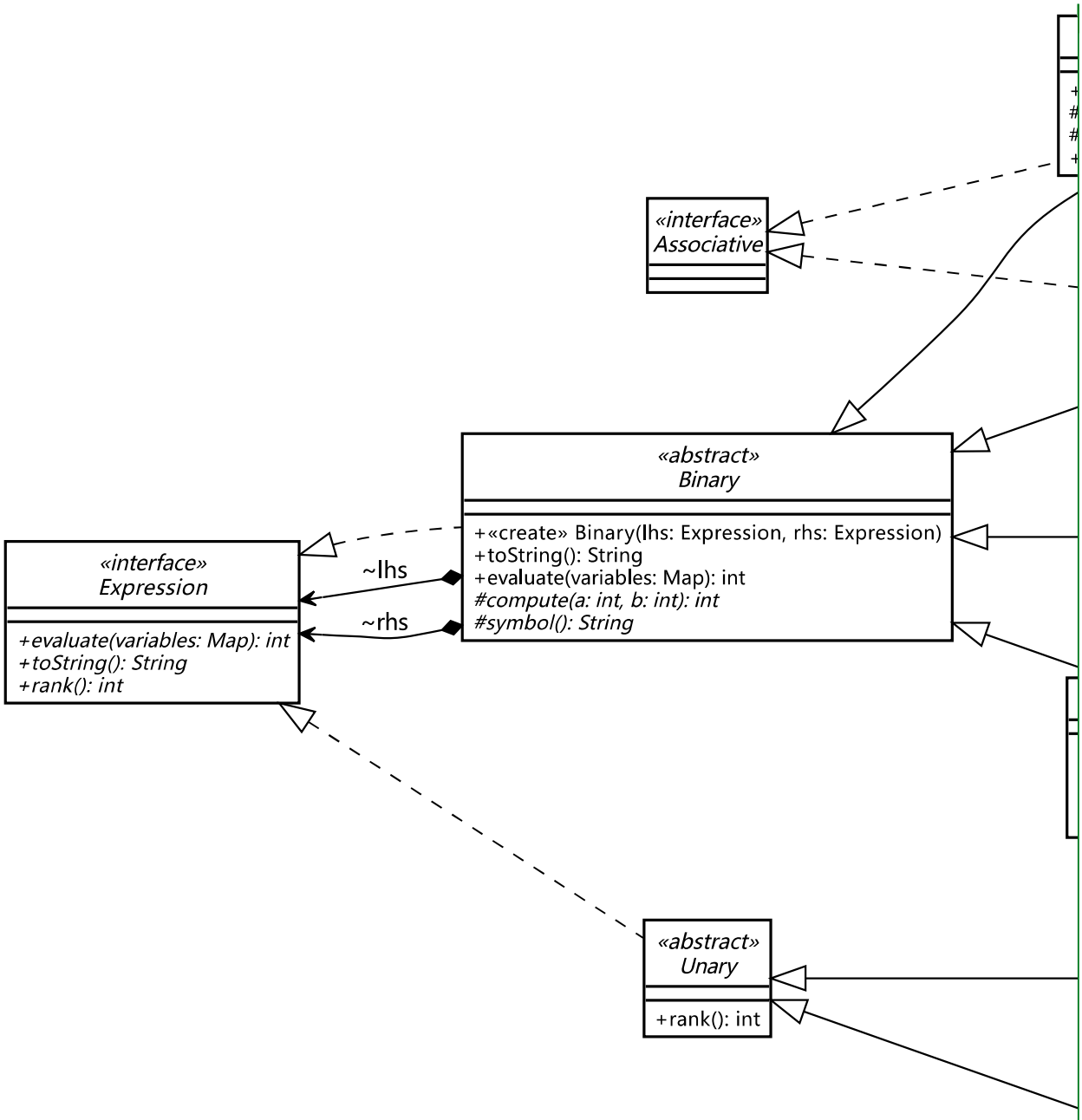
- Das Interface `Expression` schreibt folgende Methoden vor:
  - `evaluate` nimmt eine `Map<String, Integer>`, die Variablennamen Werte zuweist, und Wertet den AST mit dieser Variablenbelegung zu einem Integer aus.
  - `toString` serialisiert den AST als geklammerten Ausdruck. Dabei sollen Klammern nur genau dann ergänzt werden, wenn diese notwendig sind.
  - `rank` liefert den Rang des Elements im AST im Sinne der Präzedenz der Operatoren zurück. Werte und Variablen haben dabei einen Rang von 0, Multiplikation und Division einen Rang von 1 und Addition und Subtraktion einen Rang von 2.
- Die abstrakte Klasse `Binary` implementiert `Expression` und fungiert als gemeinsame Oberklasse von binären Operationen. Sie wird insbesondere zur Vermeidung von Code-Duplikation verwendet.
- Die Klassen `Multiplication`, `Addition`, `Division` und `Subtraction` erben von `Binary` und implementieren entsprechende Methoden, um die Implementierungen in `Binary` auf die konkreten Operationen zu konfigurieren.
- Die abstrakte Klasse `Unary` implementiert `Expression` und fungiert dual zu `Binary` für unäre Elemente des ASTs.
- Die Klassen `Value` und `Variable` erben von `Unary` und repräsentieren konkrete Werte bzw. Variablen. Ein Wert wird dabei als Integer gespeichert. Negative Zahlen werden direkt als negativer Integer gespeichert. Ein unäres Minus wird im AST nicht unterstützt. Eine Variable besteht im AST nur aus einem Bezeichner, der als String gespeichert wird.
- Das Marker-Interface `Associative` wird von `Multiplication` und `Addition` implementiert, um anzuzeigen, dass diese beiden Operationen assoziativ sind. In der String-Serialisierung müssen also keine Klammern ergänzt werden, um die Reihenfolge der Operanden einzuhalten. Beachte dazu im obigen Beispiel die unterschiedliche Klammerung bei der Ausgabe von `add` und `sub`. Durch das Marker-Interface und die Methode `rank` kann die Klammerung bei der String-Serialisierung vollständig in der abstrakten Klasse `Binary` implementiert werden.

- Die Exception `CalcException` erbt von `RuntimeException` und soll geworfen werden, wenn eine Variable ausgewertet wird, ohne dass ein Wert für diese Variable in der übergebenen Variablenbelegung enthalten ist.

Erstellen Sie zu dieser Datenstruktur folgende Diagramme:

1. Erstellen Sie ein Klassendiagramm, in dem alle Schnittstellen und Klassen des Syntaxbaums dargestellt werden: `Expression`, `Binary`, `Unary`, `Addition`, `Subtraction`, `Multiplication`, `Division`, `Associative`, `Value` und `Variable`. Stellen Sie Klassen der Standard-Library dabei nicht als eigene Klassen dar und lassen Sie Typparameter bei generischen Typen weg, d.h. geben Sie den Typ `Map<String, Integer>` als `Map` an. (3 Punkte)

   ▼ **Lösungsvorschlag**



▶ Quelltext des Diagramms

Die abstrakte Klasse `Binary` wird laut Aufgabenstellung insbesondere zur Vermeidung von Code-Duplikation verwendet. Sie hat in obigem Diagramm dazu die abstrakten Methoden `compute` und `symbol`, die in der Aufgabenstellung nicht beschrieben werden.

1. Die Methode `compute` nimmt zwei Integer und gibt einen Integer zurück. Sie wird von den einzelnen binären Operatoren mit deren mathematischer Operation überschrieben, damit die rekursive Methode `evaluate` generisch durch die abstrakte Klasse `Binary` implementiert werden kann.

2. Die Methode `symbol` gibt die String-Serialisierung eines Operators zurück, also zum Beispiel `+` für die Addition. Auf diese Weise kann die Methode `toString` genauso wie `evaluate` generisch in der abstrakten Klasse `Binary` implementiert werden.

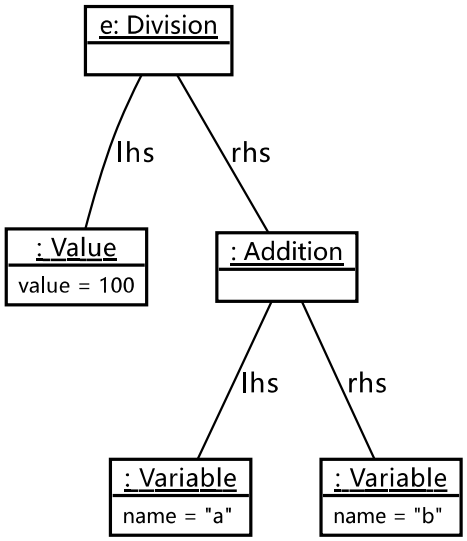
Hier sind natürlich auch andere Modellierungen möglich. So könnten insbesondere die Methoden `toString` und `evaluate` auch von allen binären Operatoren einzeln implementiert werden, auch wenn die Implementierungen dann jeweils sehr ähnlich wären.

2. Es sei die Expression `e` gegeben durch:

```
Expression e = new Division(new Value(100),
        new Addition(new Variable("a"), new Variable("b")))
System.out.println(e); // 100 / (a + b)
```

Erstellen Sie ein Objektdiagramm, in dem die an `e` beteiligten Instanzen der Klassen aus der vorherigen Teilaufgabe dargestellt werden. (2 Punkte)

▼ Lösungsvorschlag



▶ Quelltext des Diagramms

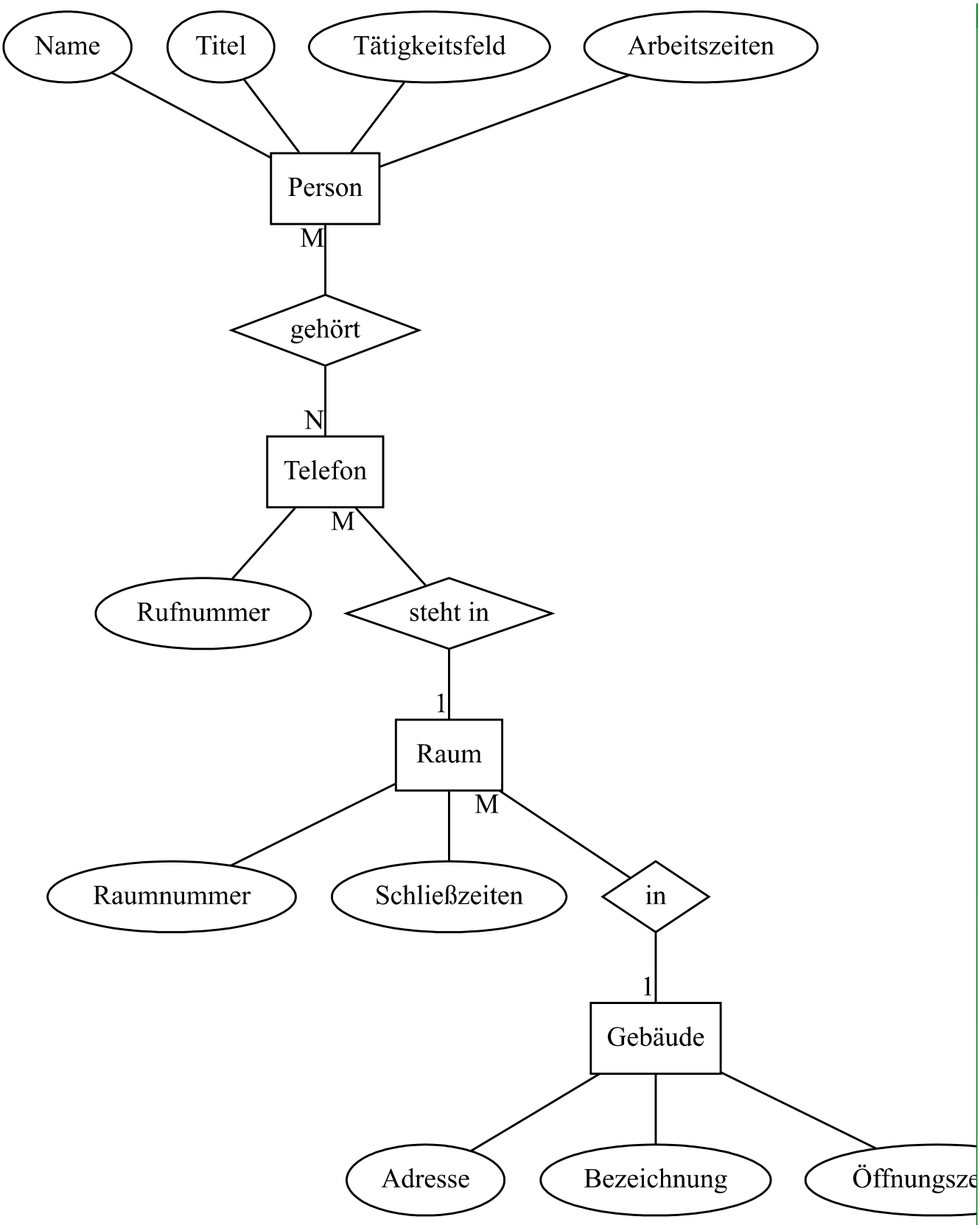
Aufgabe 8.2: ER-Diagramm

3 Punkte, leicht

Stellen Sie folgende Beschreibung eines Telefonbuchs als ER-Diagramm dar:

> Zu jedem Telefon ist eine zugehörige Rufnummer hinterlegt. Ein Telefon befindet sich immer in einem bestimmten Raum, ein Raum in einem Gebäude. Über einen Raum sind die Raumnummer und die Schließzeiten bekannt. Ein Gebäude verfügt über eine Adresse und eine Bezeichnung, sowie ebenfalls Öffnungszeiten. Ein Telefon kann, muss aber nicht, mit einer Person verknüpft sein. Andererseits können auch mehrere Personen das gleiche Telefon nutzen. Zu Personen werden Name, Titel, Tätigkeitsfeld, Arbeitszeiten gespeichert.

▼ Lösungsvorschlag



▶ Quelltext des Diagramms

Aufgabe 8.3: Programmablaufpläne und Nassi-Shneiderman-Diagramme

4 Punkte, leicht

Betrachten Sie folgendes Java-Programm:

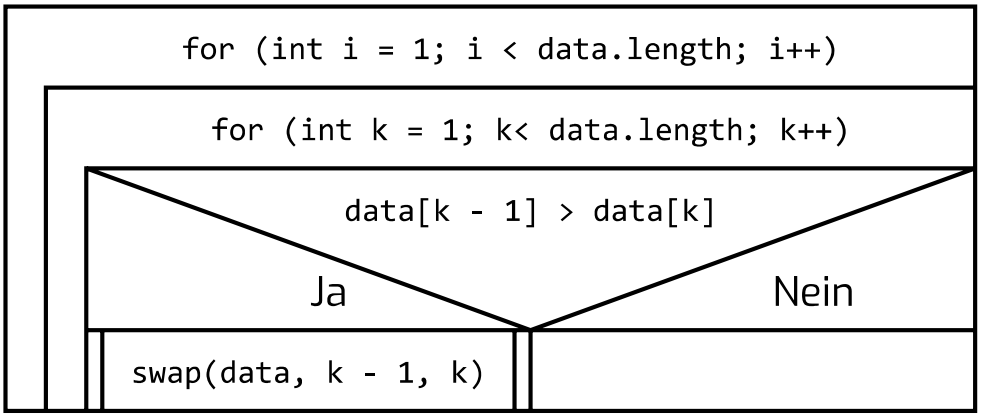
```
public class Main {
  public static void magic(int[] data) {
    for (int i = 1; i < data.length; i++) {
      for (int k = 1; k < data.length; k++) {
        if (data[k - 1] > data[k]) {
          swap(data, k - 1, k);
        }
      }
    }
  }

  private static void swap(int[] data, int a, int b) {
    int tmp = data[a];
    data[a] = data[b];
    data[b] = tmp;
  }

  public static void main(String[] args) {
    int[] data = new int[]{5, 7, 3};
    magic(data);
    for (int i = 0; i < data.length; i++) {
      System.out.print(data[i] + " ");
    }
  }
}
```

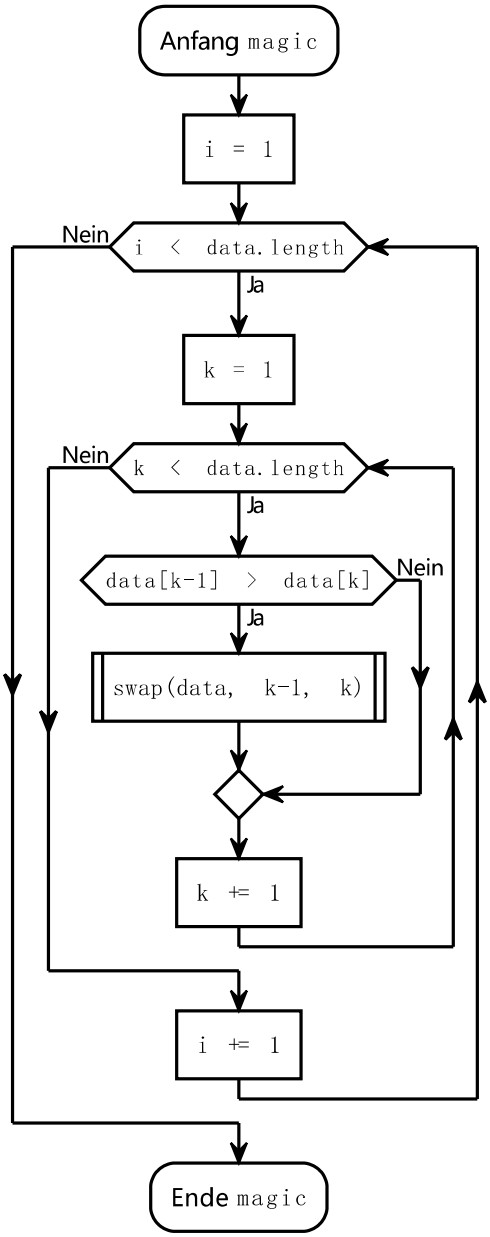
1. Zeichnen Sie ein Struktogramm, das den Code der Methode `magic` wiedergibt. Verwenden Sie die Java-Syntax zur Beschriftung. (1 Punkt)

▼ Lösungsvorschlag



2. Zeichnen Sie einen Programmablaufplan, der den Code der Methode `magic` wiedergibt. Nutzen Sie lediglich die Notation für Sequenzen, Auswahl, Algorithmusanwendung, Anfang und Ende. Verwenden Sie die Java-Syntax zur Beschriftung. (1 Punkt)

▼ **Lösungsvorschlag**



▶ Quelltext des Diagramms

3. Geben Sie die Ausgabe des Programms an. (0,5 Punkte)

▼ **Lösungsvorschlag**

```
3 5 7
```

4. Bestimmen Sie die Zeitkomplexität der Methode `magic`, in dem Sie die Anzahl der durchgeführten Vergleiche in Zeile 5 in Abhängigkeit von der Länge n des Arrays `data` als Funktion f angeben. (0,5 Punkte)

▼ **Lösungsvorschlag**

$f : \mathbb{N} \to \mathbb{N}, n \mapsto (n-1)^2$

5. Geben Sie die asymptotische Komplexität der Funktion f unter Verwendung des Landau-Symbols $\mathcal{O}$ an. (0,5 Punkte)

   ▼ Lösungsvorschlag

   $f \in \mathcal{O}(n^2)$

6. Was tut die Funktion `magic`? (0,5 Punkte)

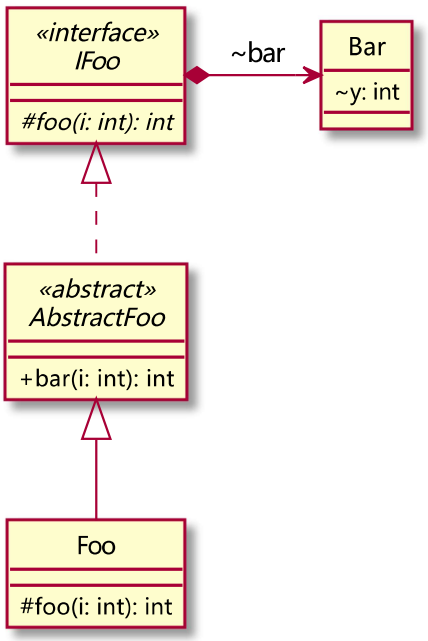
   ▼ Lösungsvorschlag

   Die Funktion `magic` sortiert das Array mit Bubblesort.

Hinweise zum Erstellen der Diagramme

UML-Klassendiagramm

UML-Klassendiagramme lassen sich sehr elegant mit PlantUML erstellen, was in CodiMD (oder HedgeDoc) direkt eingebunden ist.



▶ Quelltext des Diagramms

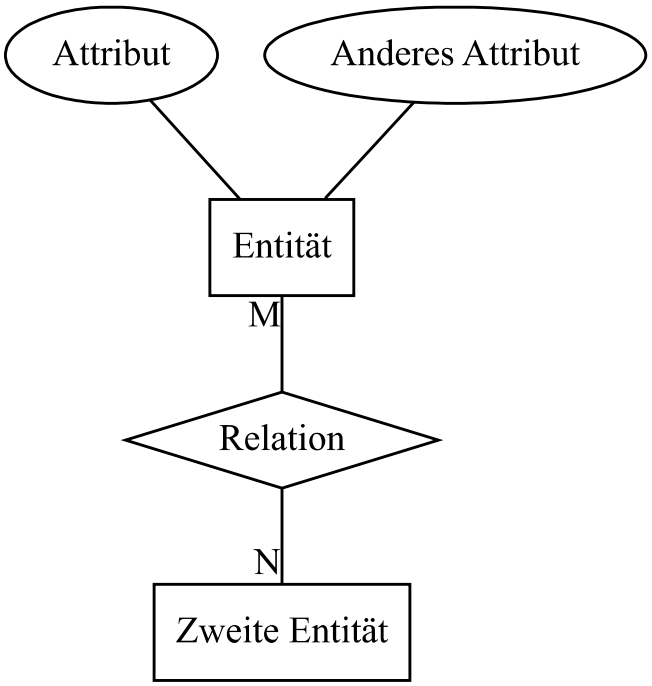
UML-Objektdiagramm

Genau wie Klassendiagramme können auch Objektdiagramme sehr elegant mit PlantUML erstellt werden.

▶ Quelltext des Diagramms

ER-Diagramme

ER-Diagramme können leicht mit leicht mit GraphViz gesetzt werden. GraphViz ist in PlantUML enthalten, was wiederum in CodiMD (oder HedgeDoc) direkt eingebunden ist.



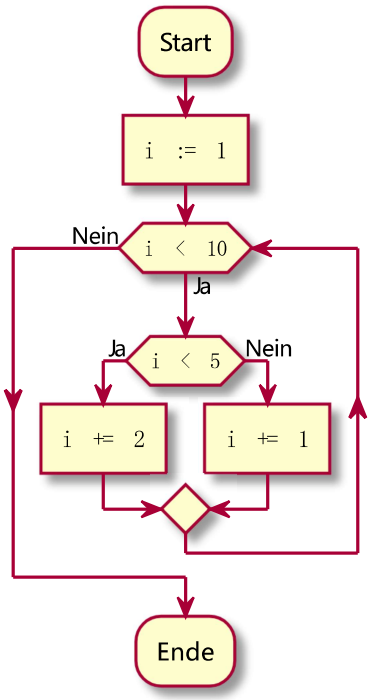
▶ Quelltext des Diagramms

Nassi-Shneiderman-Diagramme

Für das Zeichnen von Struktogramme habe ich leider keine gute Empfehlung. Ich zeichne diese, in dem ich den entsprechenden SVG- oder Ti*k*Z-Code manuell schreibe. Das ist explizit keine Empfehlung. Es gibt allerdings diverse Online-Tools, die einen bei der Erstellung von Struktogrammen aus Pseudo-Code unterstützen.

Programmablaufpläne

Programmablaufpläne können wieder gut mit PlantUML erstellt werden.



▶ Quelltext des Diagramms