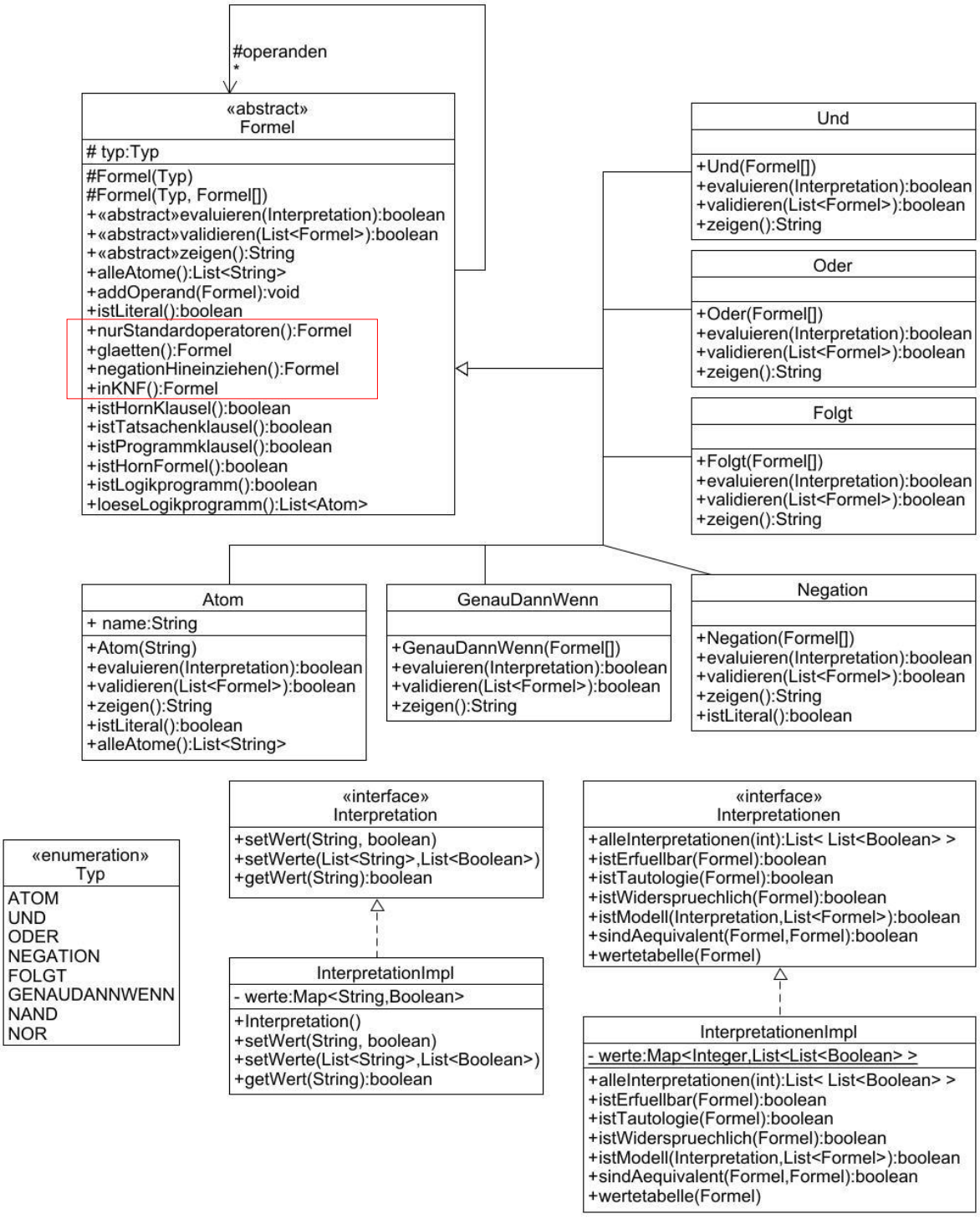
Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

Logik

Sommersemester 2022
3. Aufgabenblatt



Das Klassendiagramm zeigt den vollständigen Aufbau nach drei Aufgabenblättern, aktuell zu erstellende Teile sind mit einem roten Kasten markiert. Für Freunde präziser Klassendiagramme: <<nutzt>>-Beziehungen sind nicht eingezeichnet, weiterhin werden existierende und nutzbare get- und set-Methoden weggelassen.

Aufgabe 4 (9 Punkte)

In dieser Aufgabe werden Sie Formeln transformieren, wodurch aus einer Formel eine andere wird. Generell ist so ein Umbau einer Formel problemlos machbar, es könnten aber

Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

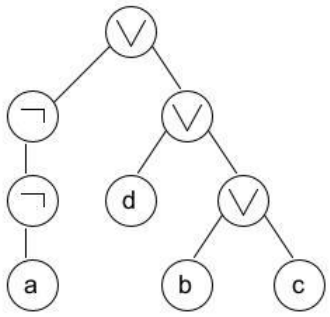
Logik

Sommersemester 2022
3. Aufgabenblatt

Seiteneffekte durch Referenzen auftreten, wenn eine Teilformel mehrfach verwandt wird. Ein einfacher Ansatz ist es, die umzuwandelnde Formel vor der Umwandlung zu clonen und dann diesen Clone zu nutzen.

Organisatorischer Hinweis: Generell ist es sinnvoll Aufgabenteil f) zusammen mit den anderen zu bearbeiten. Da der erwartete Arbeitsaufwand für dieses Aufgabenblatt höher als der für das folgende Aufgabenblatt ist, kann f) auch eine Woche danach nachgereicht werden.

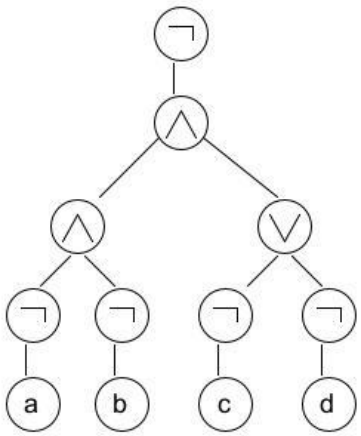
a) Realisieren Sie die Methode *nurStandardoperatoren()* für Formeln, die in der Klasse *Formel* nur angefangen wurde. Die Methode soll eine äquivalente Formel liefern, die nur Und, Oder und Negation enthält. Setzen Sie dies induktiv um, indem Sie *nurStandardoperatoren()* für Formeln basierend auf *Folgt* und *GenauDannWenn* in den zugehörigen Klassen umsetzen. Am Ende müssen zumindest die Tests der Klasse *test.NurStandardoperatorenTest* laufen.



b) Gegeben sei der Ableitungsbaum einer Formel auf der rechten Seite. Glätten Sie diese Formel von Hand (Erinnerung: keine syntaktisch unnötigen Operatoren, wie z. B. zwei Negationen hintereinander oder ein Und, das wiederum ein Und enthält) und zeichnen Sie jeden Zwischenschritt auf.

c) Realisieren Sie die Methode *glaetten()* für Formeln, die in der Klasse *Formel* nur angefangen wurde. Setzen Sie diese Methode um, so dass es keine unnötigen Schachtelungen von Und, Oder und doppelten Negationen gibt. Am Ende müssen zumindest die Tests der Klasse *test.GlaettenTest* laufen.

d) Gegeben sei der Ableitungsbaum einer Formel auf der rechten Seite. Bringen Sie diese Formel von Hand in die konjunktive Normalform und zeichnen Sie jeden Zwischenschritt auf.



e) Realisieren Sie die Methode *negationHineinziehen()* für Formeln, die in der Klasse *Formel* nur angefangen wurde und die die Formel so umformt, dass Negationen nur noch vor Atomen stehen (siehe Schritt 1 der KNF-Berechnung). Am Ende müssen zumindest die Tests der Klasse *test.NegationHineinziehenTest* laufen.

f) Realisieren Sie die Methode *inKNF()* für Formeln, die in der Klasse *Formel* nur angefangen wurde. Wahrscheinlich ist es sinnvoll, die Ergebnisse aus c) und e) zu nutzen. Am Ende müssen zumindest die Tests der Klasse *main.InKNFTest* laufen.

Hinweise: Die Umwandlung einer Liste in einen Array vom Typ der Liste ist etwas nervig und kann wie folgt umgesetzt werden:

```
List<Formel> uebernehmen = new ArrayList<>();
… // uebernehmen wird gefuellt
Formel[] tmp = new Formel[uebernehmen.size()]; // Hilfsarray
…   new Oder(uebernehmen.toArray(tmp)); // geht auch in einer Zeile
```