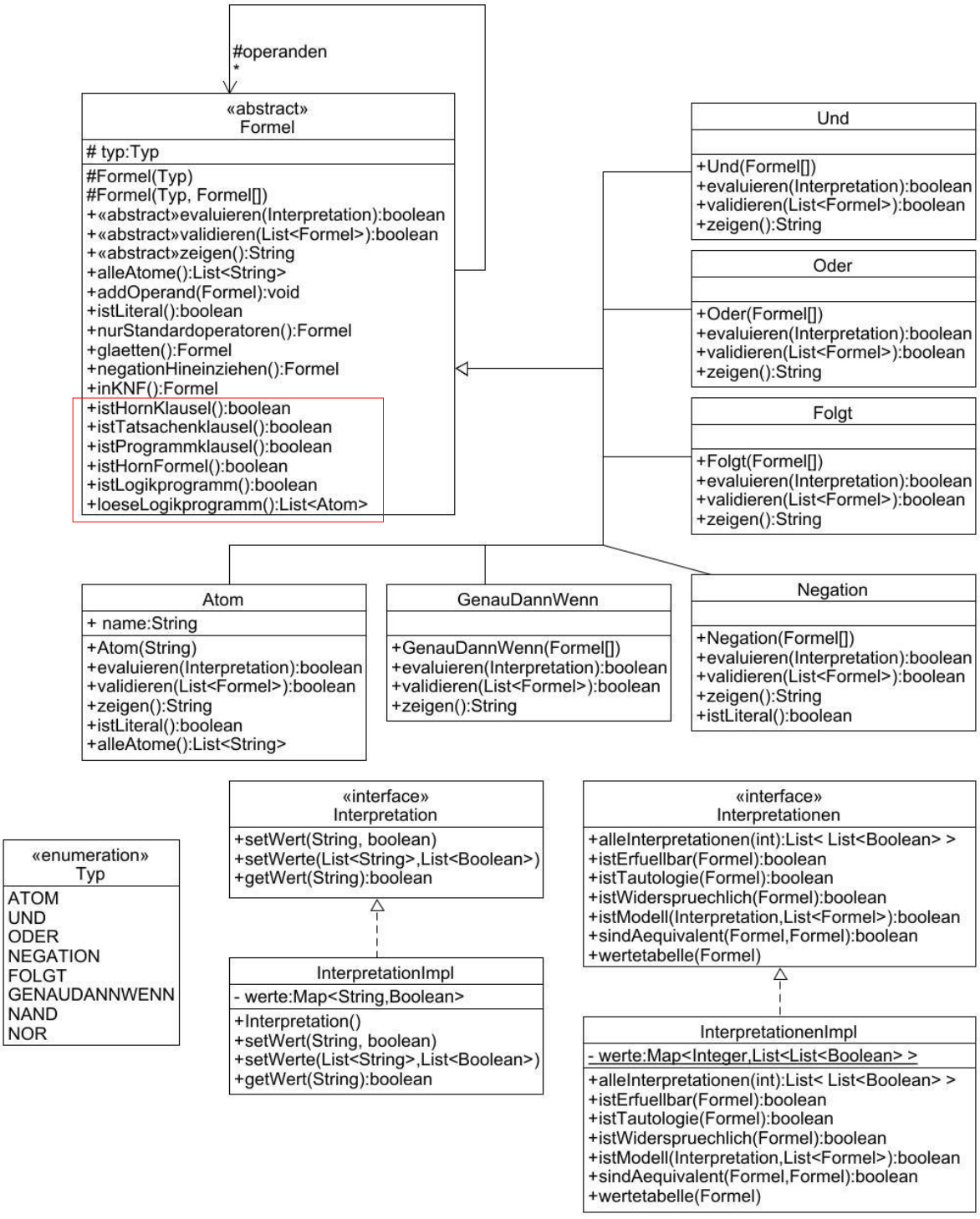
Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

Logik

Sommersemester 2023
4. 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 (Erinnerung: 5 Punkte)

f) Realisieren Sie die Methode *inKNF()* für Formeln, die in der Klasse *Formel* nur angefangen wurde. Nutzen Sie dazu den Transformationsalgorithmus aus der Vorlesung.

Wahrscheinlich ist es sinnvoll, die Ergebnisse aus c) und e) zu nutzen. Am Ende müssen zumindest die Tests der Klasse *main.InKNFTest* laufen.

Aufgabe 5 (4 Punkte)

Ziel der Aufgabe ist es mit unserem kleinen Aussagenlogik-Framework die Nutzung von Logik-Programmen umzusetzen. Dies teilt sich in folgende Teilaufgaben auf, für die alle Tests diesmal in *test.HornTest* zusammengefasst sind und wieder erfüllt sein müssen.

a) Schreiben Sie für Formeln eine Methode *istHornKlausel()* die überprüft, ob es sich bei dieser *Formel* um eine Horn-Klausel handelt. Sie können natürlich die bereits umgesetzte Prüfung *istLiteral()* nutzen.

b) Schreiben Sie für Formeln eine Methode *istTatsachenklausel()* die überprüft, ob es sich bei dieser *Formel* um eine Tatsachenklausel (Fakt) handelt.

c) Schreiben Sie für Formeln eine Methode *istProgrammklausel()* die überprüft, ob es sich bei dieser *Formel* um eine Programmklausel handelt.

d) Schreiben Sie für Formeln eine Methode *istLogikprogramm()* die überprüft, ob es sich bei dieser *Formel* um ein Logikprogramm handelt.

e) [optional, etwas aufwändig, aber sehr interessant] Schreiben Sie eine Methode *loeseLogikprogramm()*, die für ein Logikprogramm alle Atome (Tatsachenklauseln, Fakten) als Liste (*List<Atom>*) berechnet, die aus dem Logikprogramm abgeleitet werden können. Nutzen Sie dazu das inkrementelle Berechnungsverfahren aus der Vorlesung. Beachten Sie, dass die Berechnungen nach Skript rein auf syntaktischer Basis geschehen und so keine Interpretationsklassen benötigt werden.

   Für ein Logikprogramm der Form

   b
   b ⇒ a
   c ⇒ d

```
this.logikprogramm2 = new Und(
        new Atom("b")
        , new Oder(new Negation(new Atom("b")), new Atom("a"))
        , new Oder(new Negation(new Atom("c")), new Atom("d"))
);
```

   Sollte der Methodenaufruf

```
this.logikprogramm2.loeseLogikprogramm();
```

   folgendes Ergebnis in einer beliebigen Reihenfolge liefern.

```
[Atom [name=b, typ=ATOM, operanden=[]]
, Atom [name=a, typ=ATOM, operanden=[]]]
```

f) [optional] Setzen Sie das Logikprogramm aus Folie 91 mit unserem Klassensystem um und geben Sie die berechnete Lösung aus.