Karlsruher Institut für Technologie

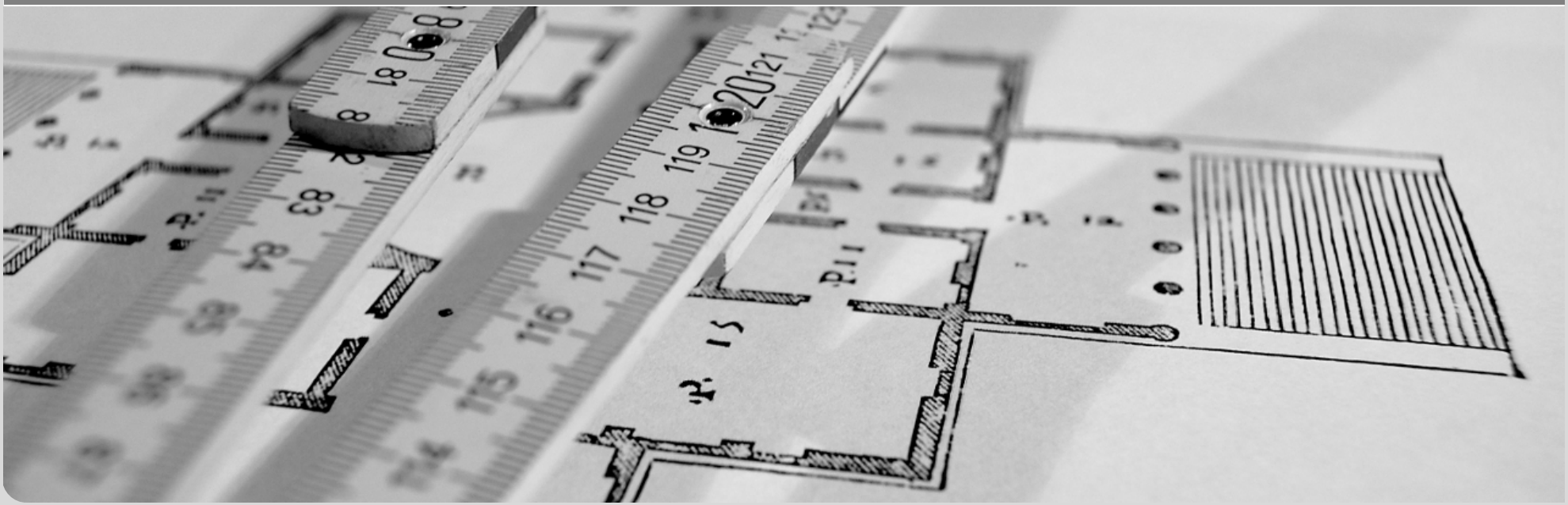
Software Design and Quality

SWT

4. Tutorium

Entwurfsmuster

Tino Fuhrmann | 16. Juni 2015



Gliederung

Varianten-Muster

Wofür werden Varianten-Muster verwendet?

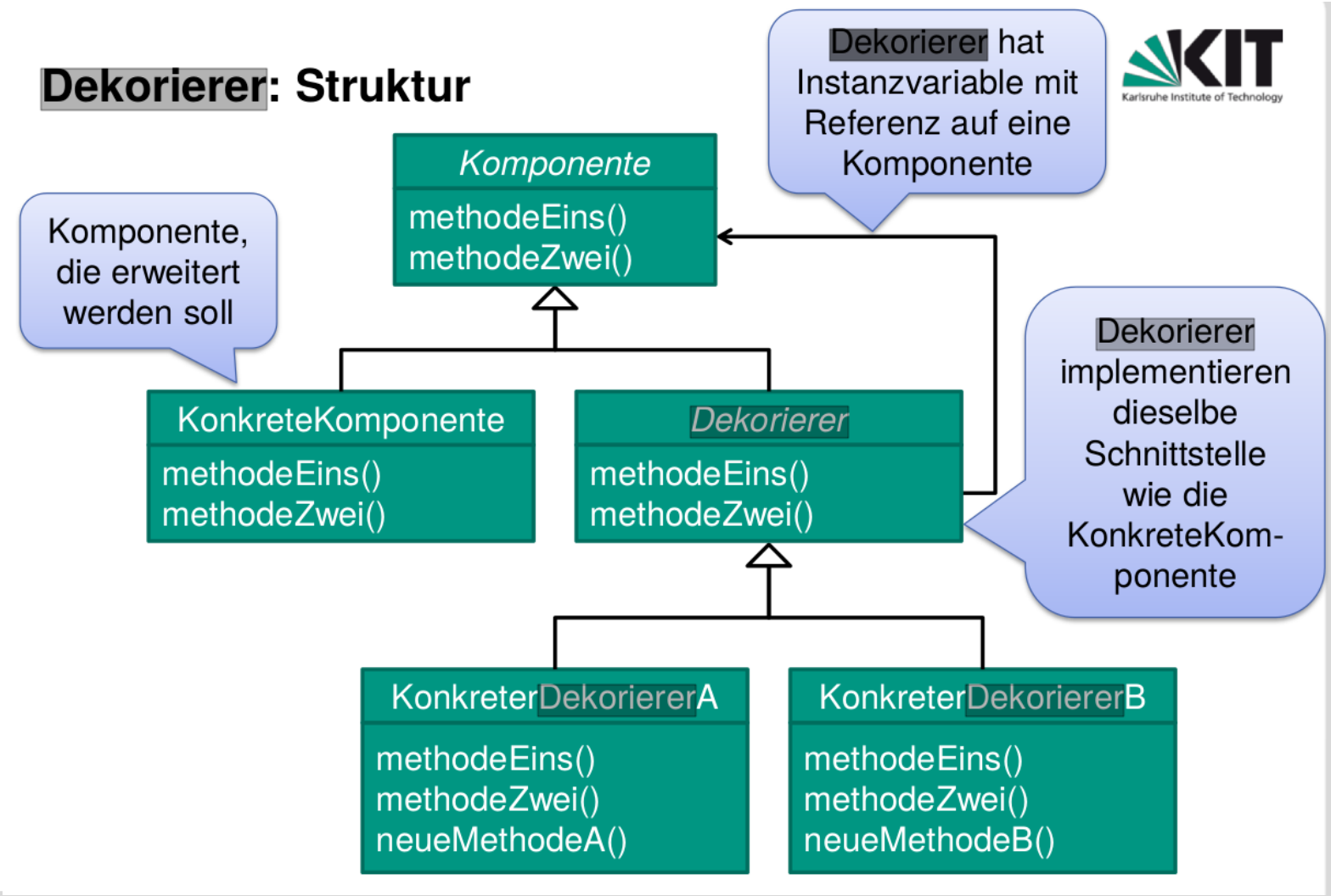
Varianten-Muster

Wofür werden Varianten-Muster verwendet?

- Gemeinsamkeit herausziehen
- Code-Wiederholungen vermeiden

Dekorierer

Fügt dynamisch neue Funktionalität zu einem Objekt hinzu. (Alternative zu Vererbung



Gruppenarbeit

- Jeder bekommt eine Karte mit einem Varianten-Entwurfsmuster.
- Innerhalb der Gruppe soll das Entwurfsmuster vorgestellt und an einem Beispiel die Funktionsweise erläutert werden
- Jetzt: Beispiel überlegen (5 Minuten)
- Gruppennummern stehen hinten auf den Karten.
- Bei Fragen und Unklarheiten: melden
- Los gehts!

Aufgabe 1: Schablonenmethode und Strategie

Welches gemeinsame Ziel haben Schablonenmethode und Strategie?
Inwiefern unterscheiden sie sich beim Erreichen dieses Ziels?

Aufgabe 1: Schablonenmethode und Strategie

Karlsruher Institut für Technologie

Welches gemeinsame Ziel haben Schablonenmethode und Strategie? Inwiefern unterscheiden sie sich beim Erreichen dieses Ziels?

- beide Erlauben es Varianten eines Algorithmus zu verwenden
- Schablonenmethode: Vererbung um Teile eines Algorithmus zu variieren
- Strategie: gesamten Algorithmus über Delegation austauschbar

Wahr oder falsch?

Eine Fabrikmethode kann eine Einschubmethode bei einer Schablonenmethode für Objekterzeugung sein.

Wahr oder falsch?

Eine Fabrikmethode kann eine Einschubmethode bei einer Schablonenmethode für Objekterzeugung sein.
wahr

Die Fabrikmethode sorgt dafür, dass nur eine einzige Instanz einer Klasse fabriziert wird.

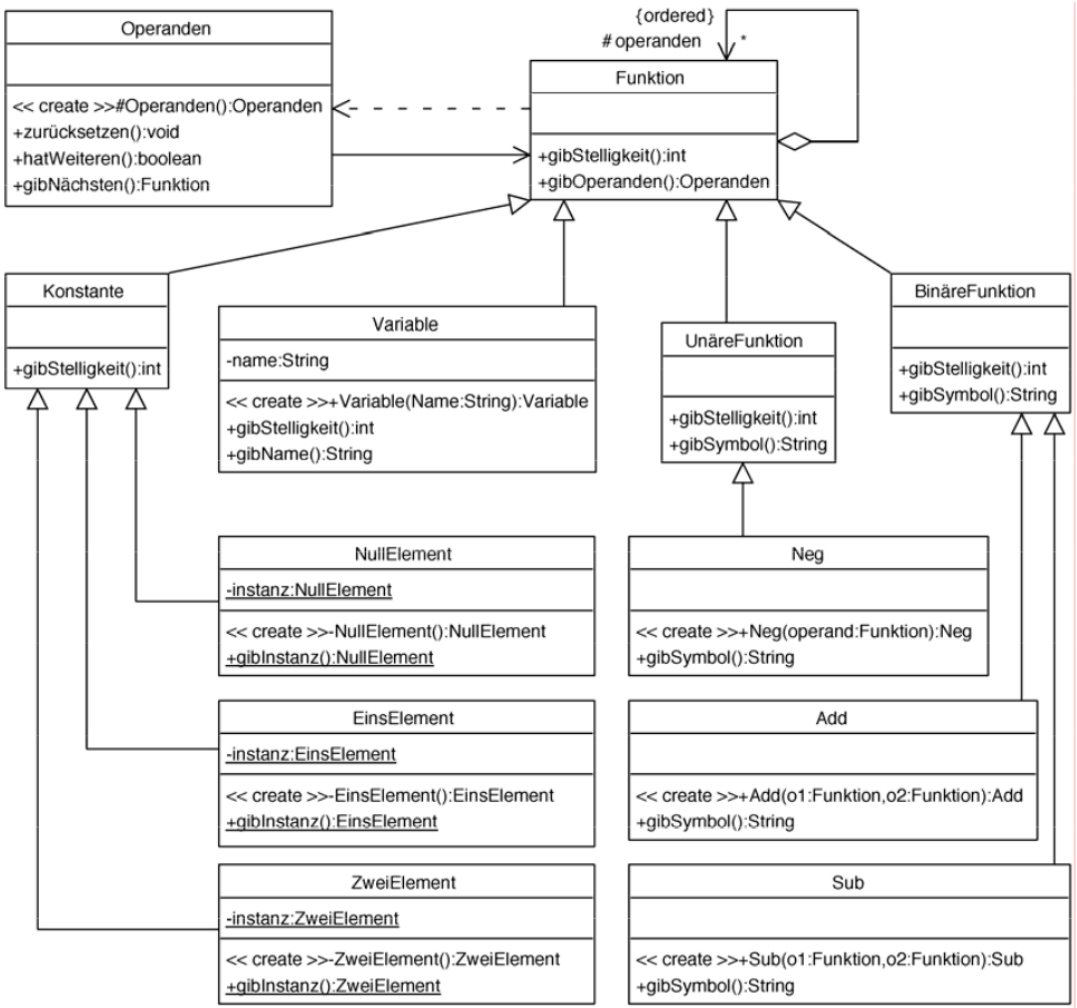
Wahr oder falsch?

Eine Fabrikmethode kann eine Einschubmethode bei einer Schablonenmethode für Objekterzeugung sein.
wahr

Die Fabrikmethode sorgt dafür, dass nur eine einzige Instanz einer Klasse fabriziert wird.
falsch

Entwurfsmuster in Klassendiagramm

Wie ist hier Iterator/Kompositum realisiert?



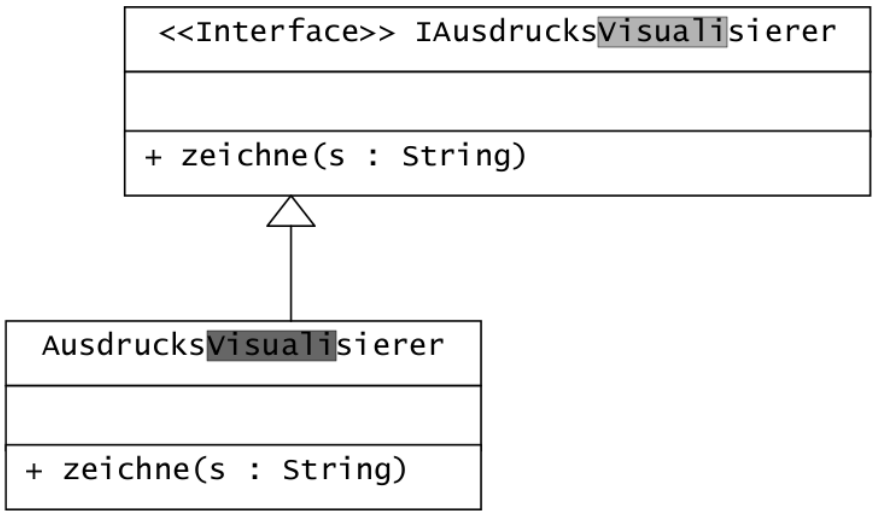
Lösung

Iterator: ja, bei „Operanden“
Neben den typischen Funktionen +hatWeiteren():boolean und +gibNächsten():Funktion sind auch die typische Assoziation und „benutzt“-Relation vorhanden. Die Methode +zurücksetzen():void ist zusätzlich.
Kompositum: ja, bei „Funktion“
„Funktion“ ist Kompositum und Komponente in einem. Die Methoden +fügeHinzu(:Komponente):void und +entferne(:Komponente):void fehlen zwar, die gemeinsame Operation +gibStelligkeit():int und die Funktion zur Rückgabe der Kinder +gibOperanden(), sowie die entsprechende Assoziation sind vorhanden.

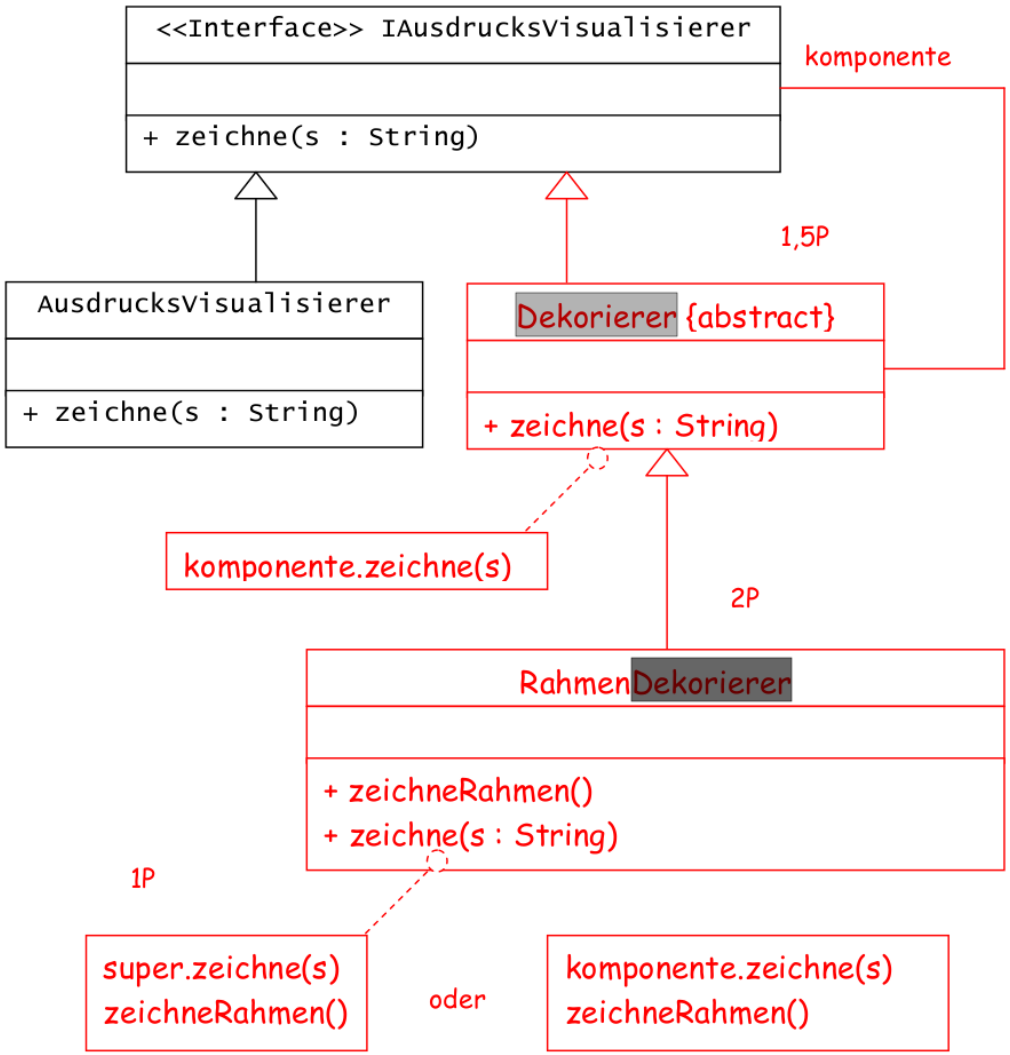
Dekorierer

d.) Die folgende Schnittstelle **IAusdrucksVisualisierer** wird von Ihrem Programm als visuelle Komponente verwendet, um arithmetische Ausdrücke auf dem Bildschirm auszugeben. Der Methode **zeichne(s : String)** übergibt man dazu den arithmetischen Ausdruck als Zeichenkette, die von dem konkreten **AusdrucksVisualisierer** dargestellt wird.

Verwenden Sie das Entwurfsmuster *Dekorierer* um einen **RahmenDekorierer** zu entwerfen. Dieser soll eine zusätzliche Methode **zeichneRahmen()** bieten, die die visuelle Komponente mit einem Rahmen dekoriert. Ergänzen Sie dazu das folgende UML-Klassendiagramm um einen abstrakten **Dekorierer** und um einen konkreten **Rahmendekorierer**. Geben Sie die Implementierung der **zeichne(s : String)** Methode in dem konkreten **Dekorierer** an (Die Implementierung der Methode **zeichneRahmen()** brauchen Sie nicht anzugeben). (4,5P)



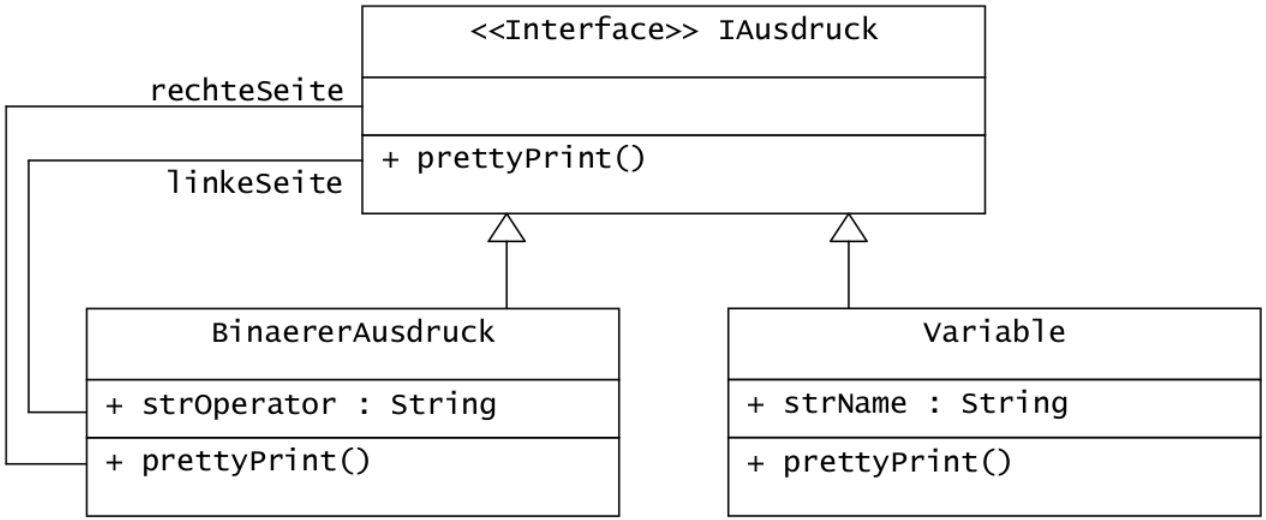
Lösung


<<Interface>> IAusdrucksVisualisierer
+ zeichne(s : String)
komponente
1,5P
AusdrucksVisualisierer
+ zeichne(s : String)
Dekorierer {abstract}
+ zeichne(s : String)
komponente.zeichne(s)
2P
RahmenDekorierer
+ zeichneRahmen()
+ zeichne(s : String)
1P
super.zeichne(s)
zeichneRahmen()
oder
komponente.zeichne(s)
zeichneRahmen()

Besucher Einführung: Aufgabe

Aufgabe 2: Entwurfsmuster (13P)

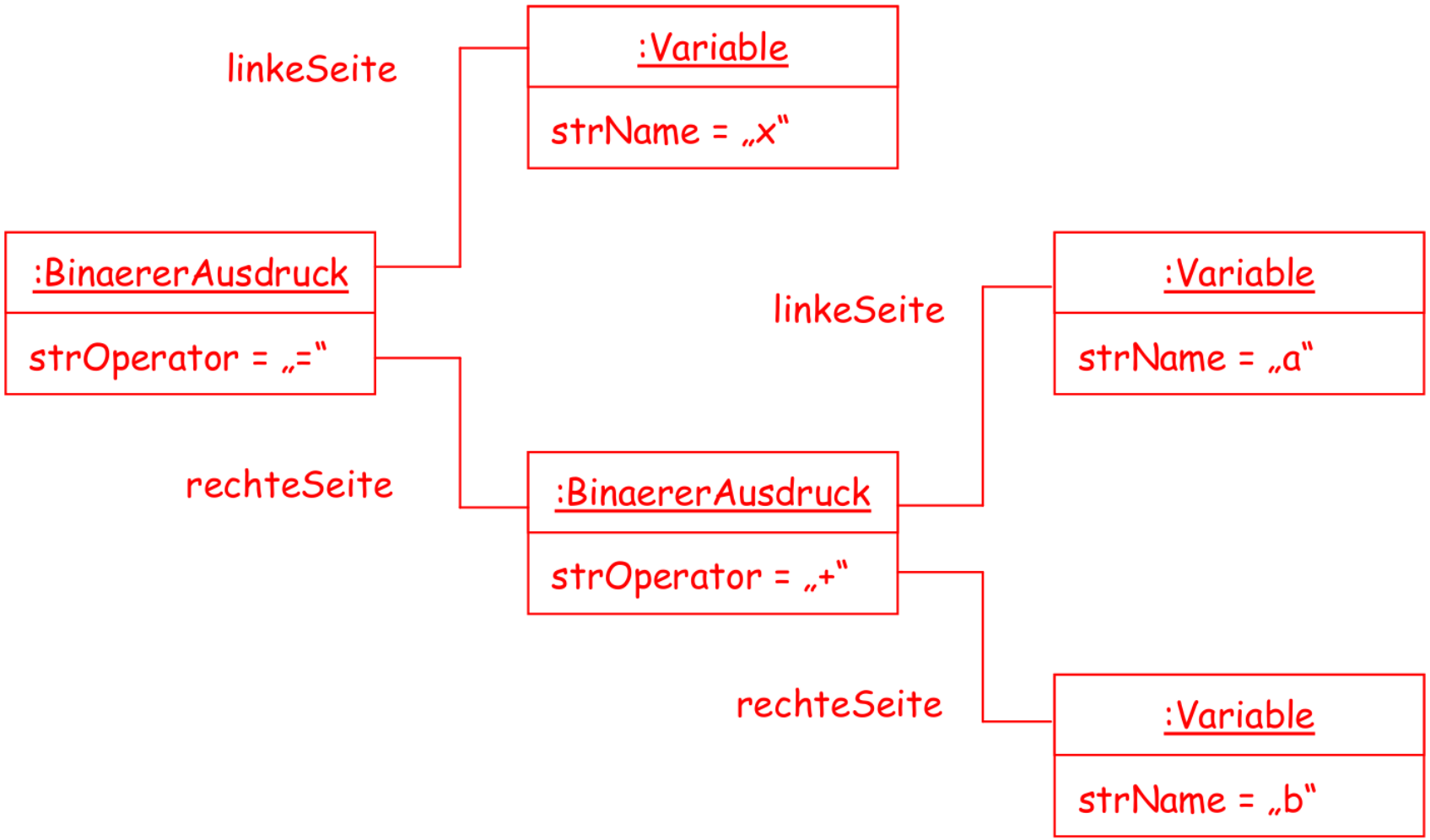
Sie möchten einen Zerteiler (Parser) für arithmetische Ausdrücke programmieren. Die folgenden Java-Klassen repräsentieren die einzelnen Bestandteile solcher arithmetischer Ausdrücke. Konkrete arithmetische Ausdrücke können somit als Baum von Instanzen dieser Klassen im Speicher dargestellt werden.



a.) Betrachten Sie den arithmetischen Ausdruck `x = a + b`. Geben Sie für diesen konkreten arithmetischen Ausdruck ein entsprechendes UML-Objektdiagramm an. Verwenden Sie die gegebenen Klassendefinitionen.

Hinweis: Sie brauchen die Objekte nicht zu benennen, sondern können anonyme Objekte verwenden. (3,5P)

Besucher Einführung: Lösung

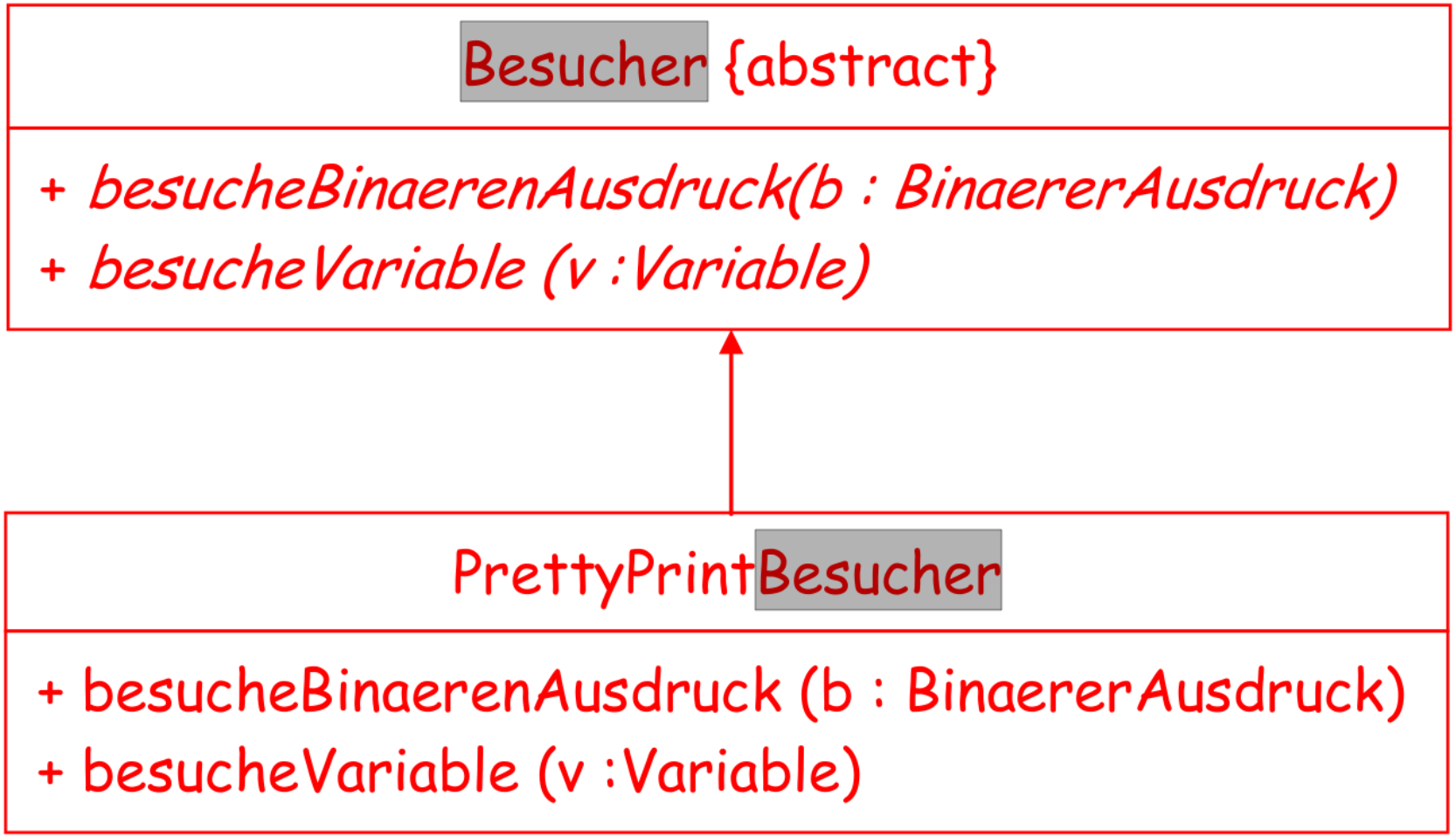


Besucher Teil 1: Aufgabe

b.) Die Klassen `BinaererAusdruck` und `Variable` weisen beide die gemeinsame Funktion `prettyPrint()` auf. Während sich die Struktur des Klassenmodells in Zukunft nicht verändern wird, sind für die weitere Entwicklung eine Reihe von neuen Operationen für die Klassen `BinaererAusdruck` und `Variable` geplant. Sie entscheiden sich daher, vorausschauend das Entwurfsmuster *Besucher* zu verwenden und die Operation `prettyPrint()` in einer Besucherklasse zu kapseln.

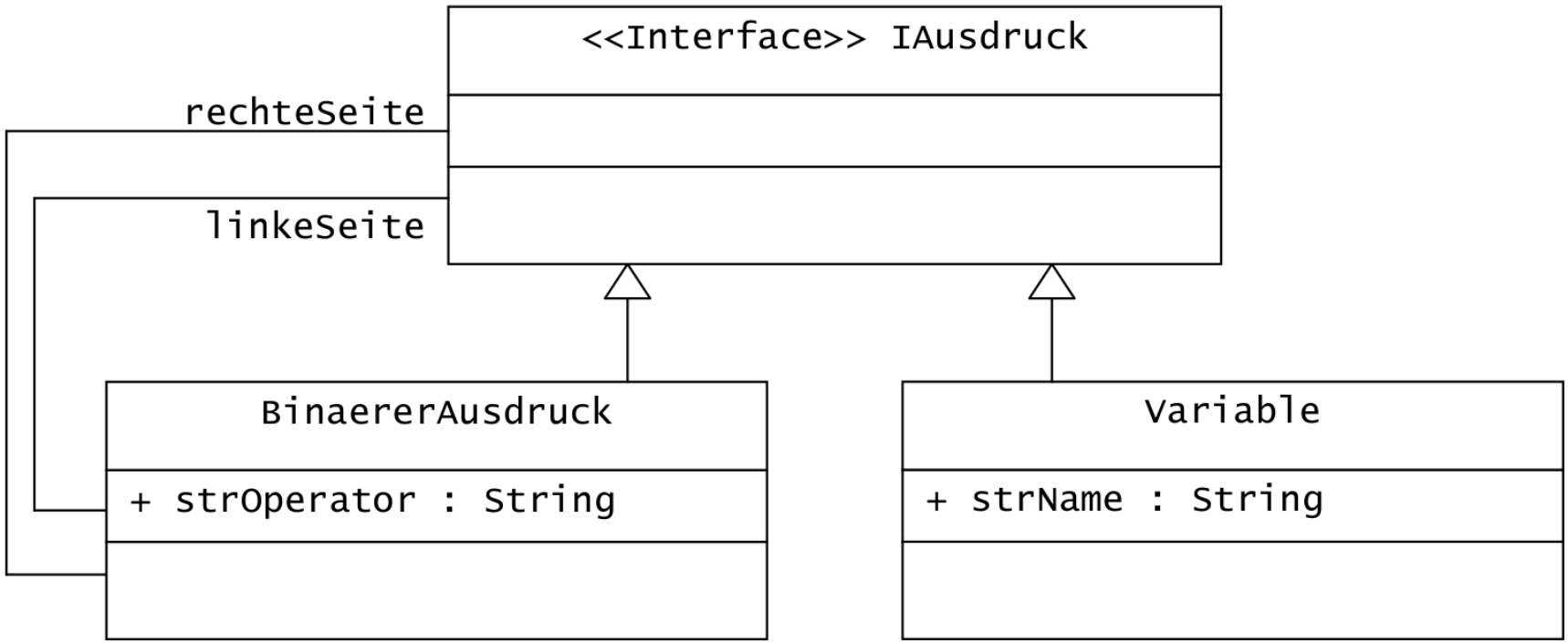
Zeichnen Sie ein UML-Klassendiagramm, das für das gegebene Klassenmodell eine abstrakte Besucherklasse und einen konkreten `PrettyPrintBesucher` enthält. Deklarieren Sie alle Methoden als „public“. (3,5P)

Besucher Teil 1: Lösung

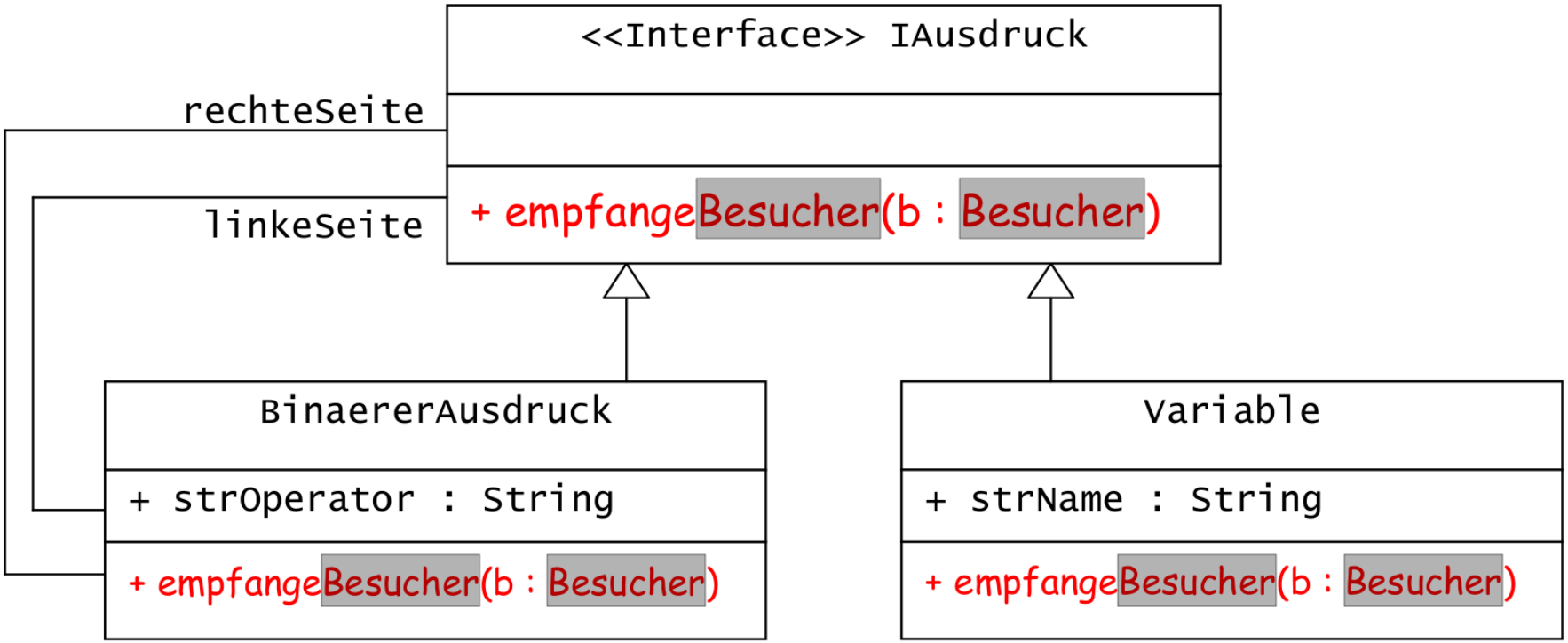


Besucher Teil 2: Aufgabe

c.) Vervollständigen Sie das folgende Klassendiagramm um Funktionen, die für das Verwenden des in Aufgabenteil b.) entworfenen Besuchers notwendig sind. Deklarieren Sie alle Methoden als „public“. (1,5P)



Besucher Teil 2: Lösung



Ende

Fragen zu Übungsblatt oder Vorlesung?

Ende

Fragen zu Übungsblatt oder Vorlesung?
Ende