

Fakultät Informatik

Professur Softwaretechnologie

OOSE 12

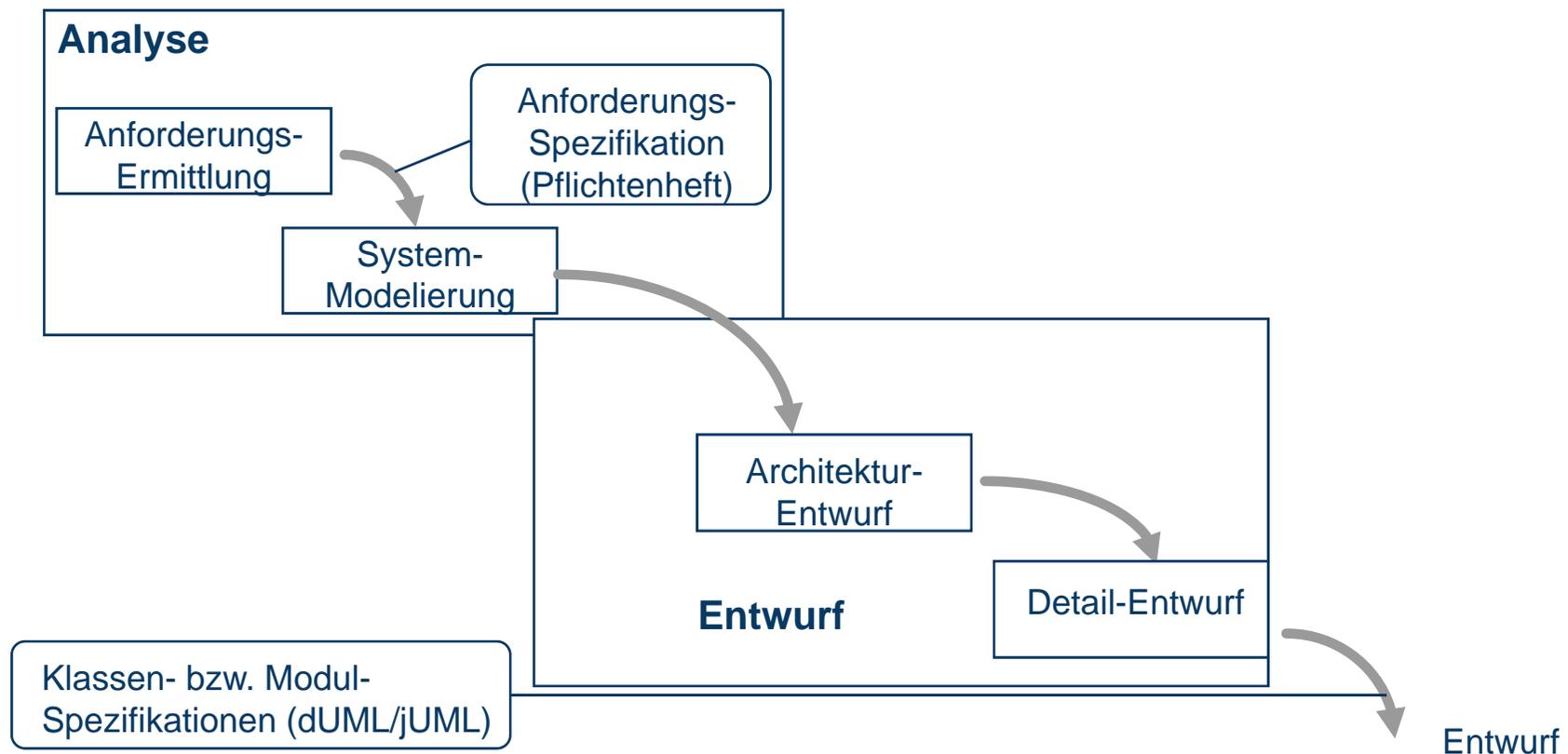
**VON DER ANALYSE ZUM ENTWURF
(KLAUSURVORBEREITUNG)**

Dr.-Ing. Birgit Demuth
Sommersemester 2019

Wiederholung

- Von Analyse (aUML) zum Entwurf (dUML) und zum Feinentwurf/Implementierung (jUML)
- Verfeinerungsoperationen im Softwareentwurf (dUML, jUML), Objektorreicherung
- [Anwendung von Entwurfsmustern zur Erhöhung der Wiederverwendbarkeit (dUML)]
- Datenstrukturentwurf/Auswahl generischer Collections (jUML)

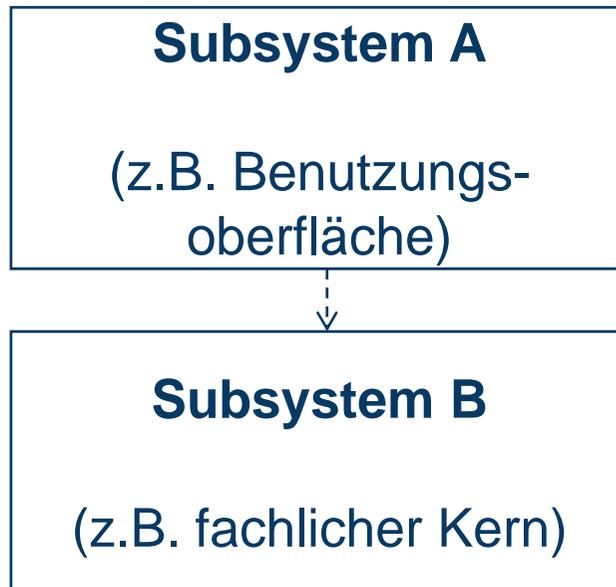
Von Analyse (aUML) zum Entwurf (dUML) und zum Feinentwurf/Implementierung (jUML)



Einige Kriterien für guten Entwurf

- Korrektheit
 - Erfüllung der Anforderungen
 - Wiedergabe aller Funktionen des Systemmodells
 - Sicherstellung der nichtfunktionalen Anforderungen
- Verständlichkeit
 - Gute Dokumentation
- Anpassbarkeit
- Hohe Kohäsion innerhalb der Komponenten
- Schwache Kopplung der Komponenten
- Wiederverwendung
- Stabilität und Zuverlässigkeit
- Angemessene Ressourcenverwendung

Hohe Kohäsion und Schwache Kopplung

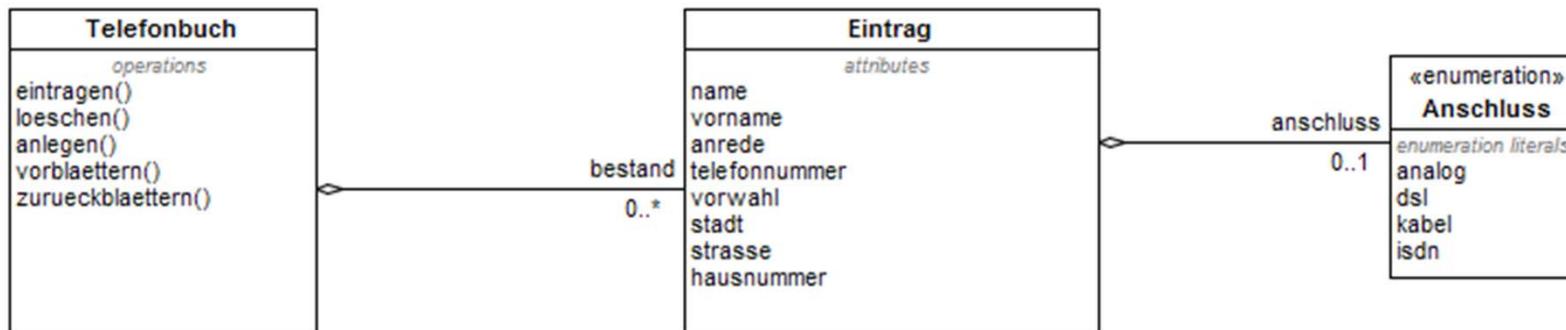


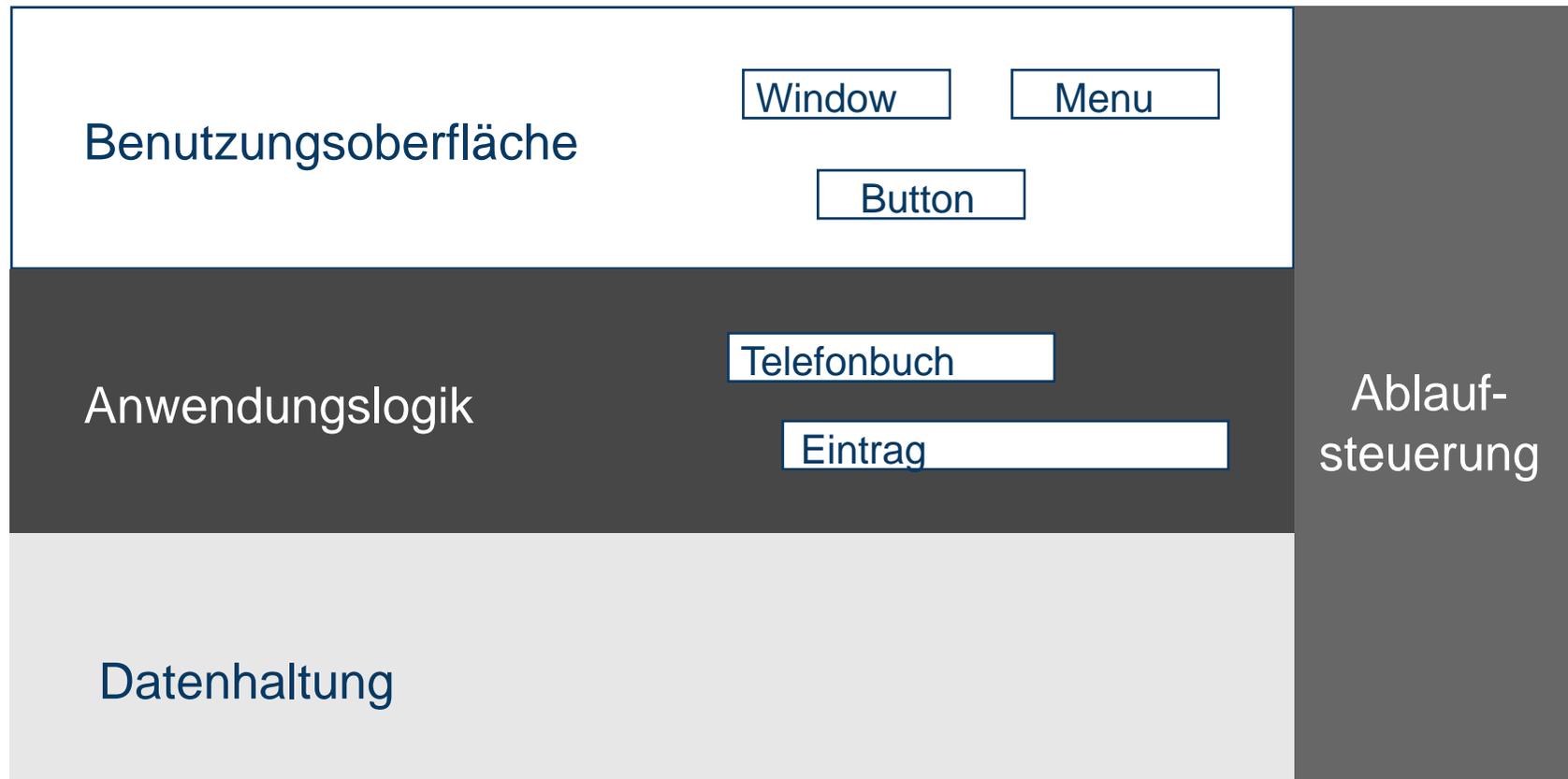
- Subsystem B darf keine Information und Funktionalität enthalten, die zum Zuständigkeitsbereich von A gehört und umgekehrt.
- Es muss möglich sein, Subsystem A weitgehend auszutauschen oder zu verändern, ohne Subsystem B zu verändern.
- Die meisten Änderungen von Subsystem B sollten nur relativ einfache Änderungen in Subsystem A nach sich ziehen.
- *Beispiele zur konkreten technischen Realisierung siehe MVC-Architektur und Entwurfsmuster*

Vom Analysemodell zum Entwurfsmodell

TELEFONBUCH

Analyse Telefonbuch



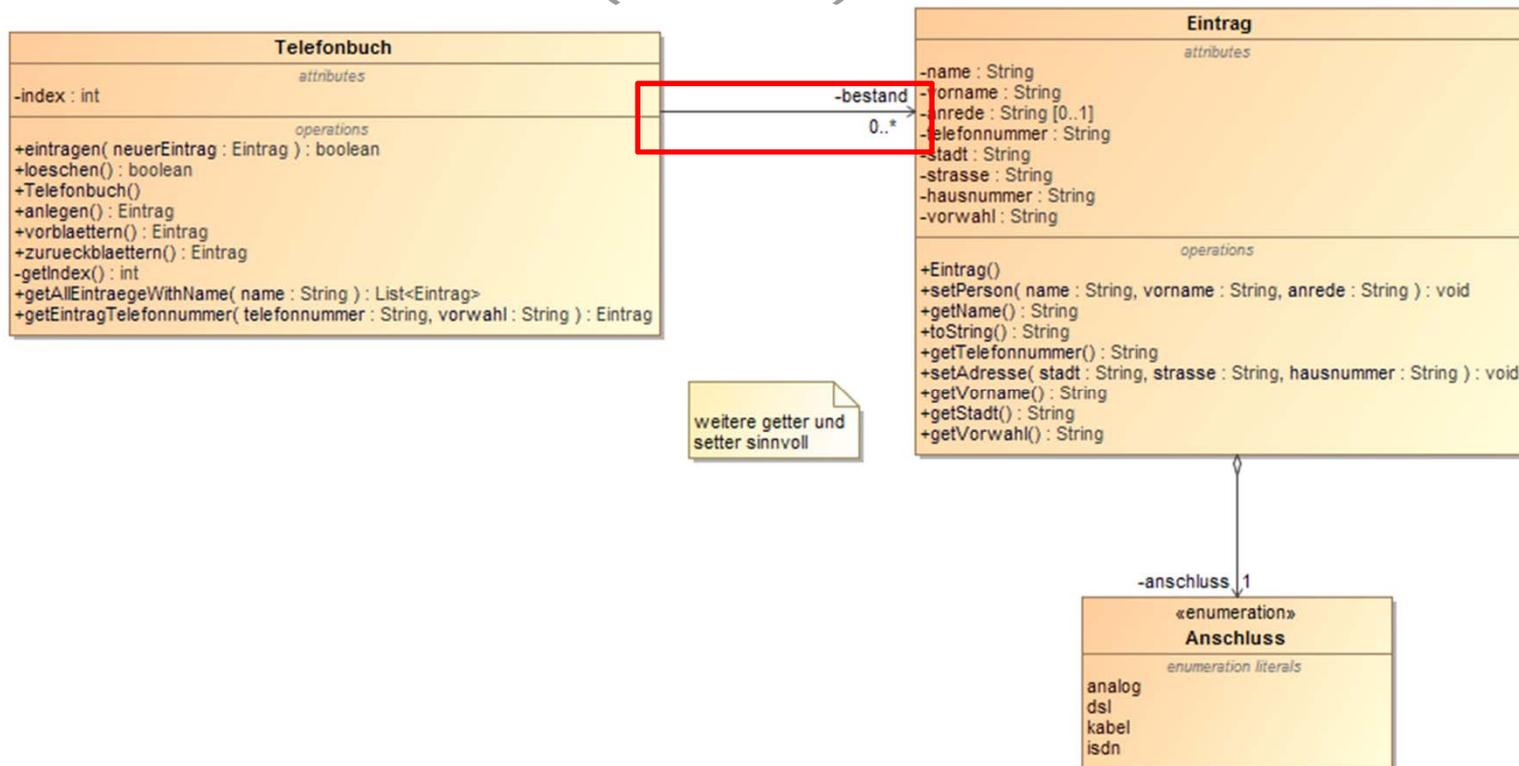


Objektanreicherung (Object Fattening)

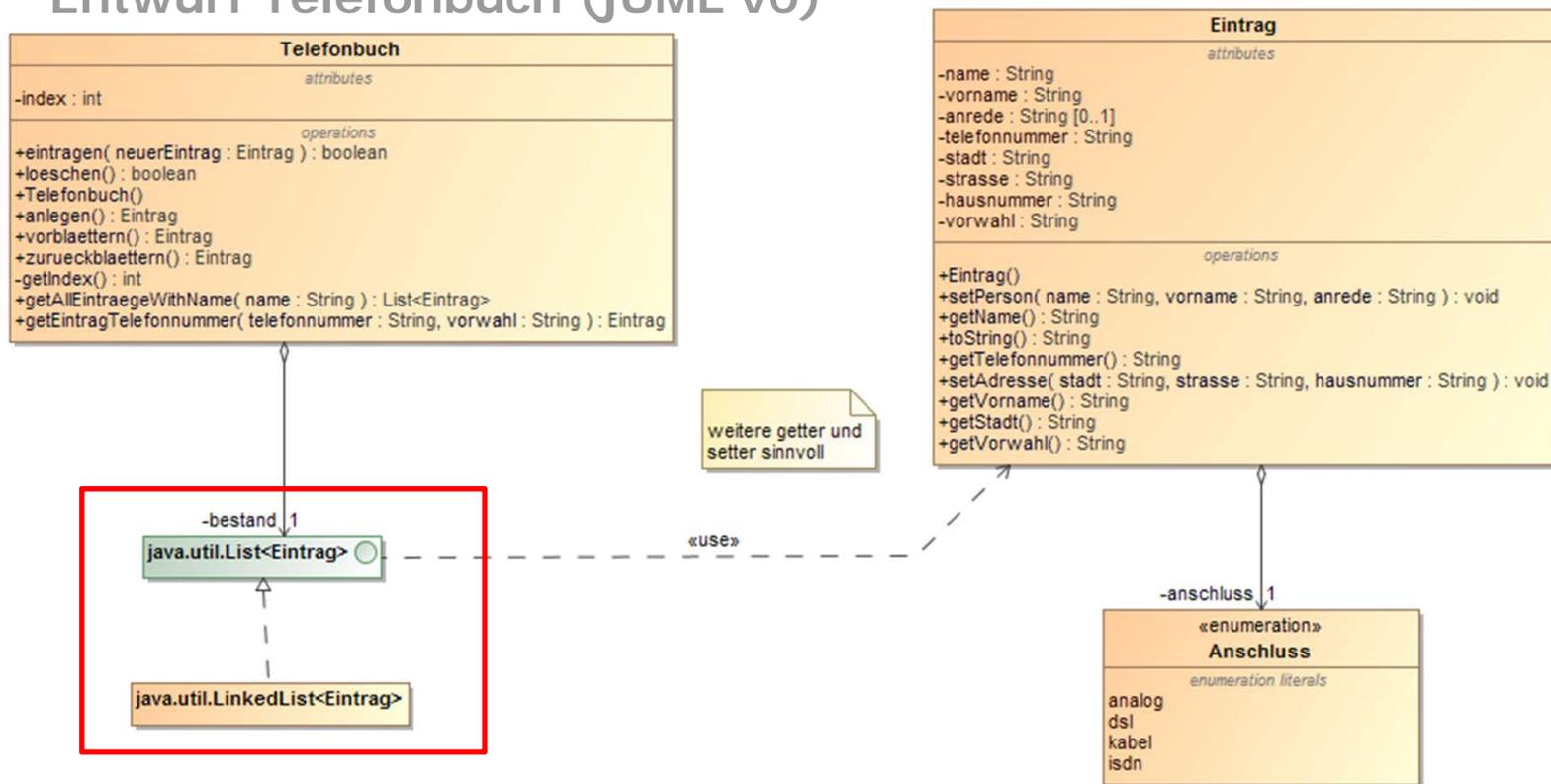
Verfeinerungsprozess zur Entwurfszeit, der

- Objekte des Domänenmodells mit Implementierungsinformationen „verfettet“
 - an ein Objekt aus dem Domänenmodell Unterobjekte anlagert
 - Unterobjekte „verfettet“
 - Beziehungen zu Plattformen, Komponentenmodellen, Frameworks, Klassenbibliotheken ... beschreibt
- Die Objekte aus dem Domänenmodell bleiben erhalten, werden aber immer „dicker“

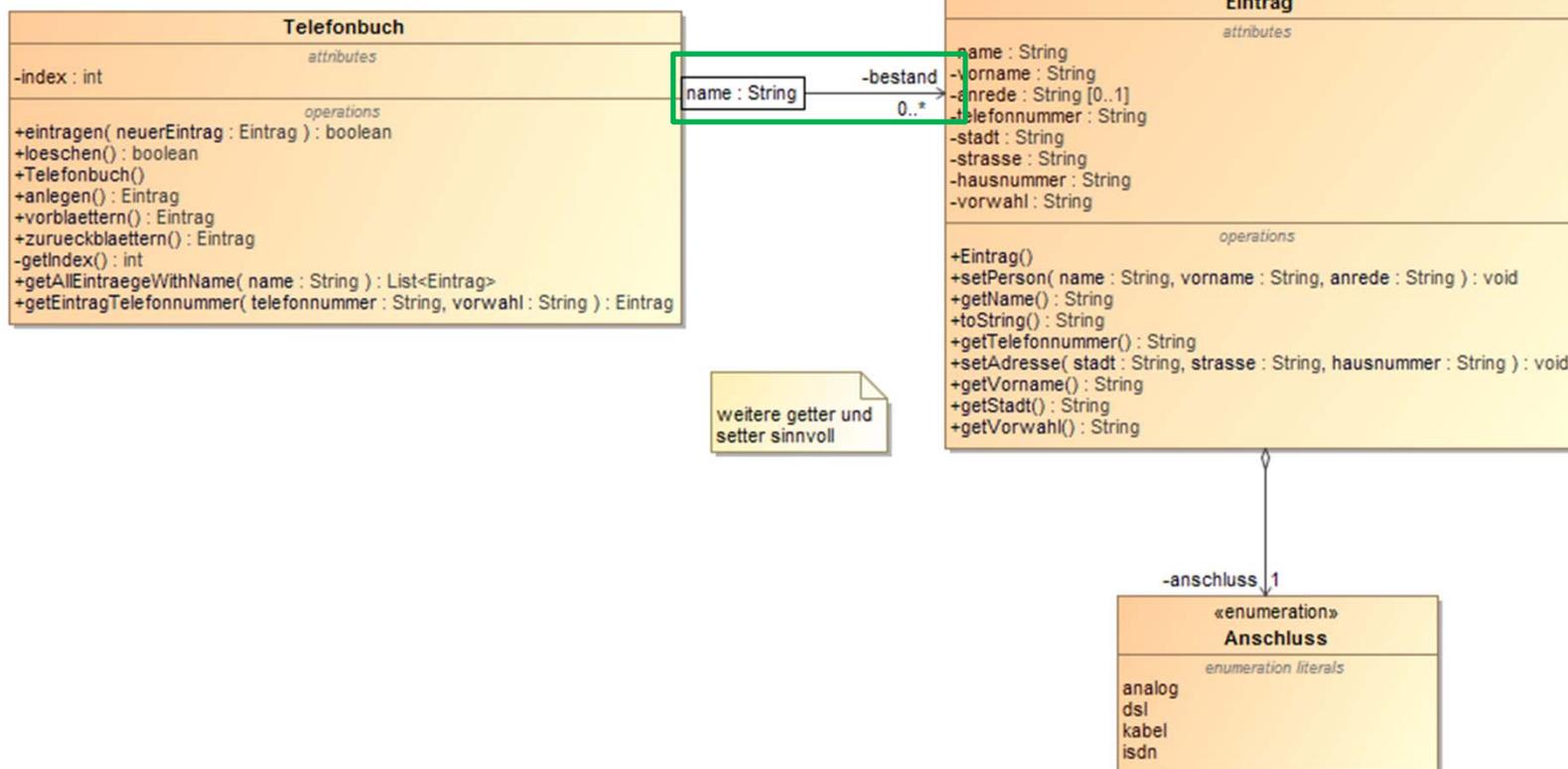
Entwurf Telefonbuch (dUML v0)



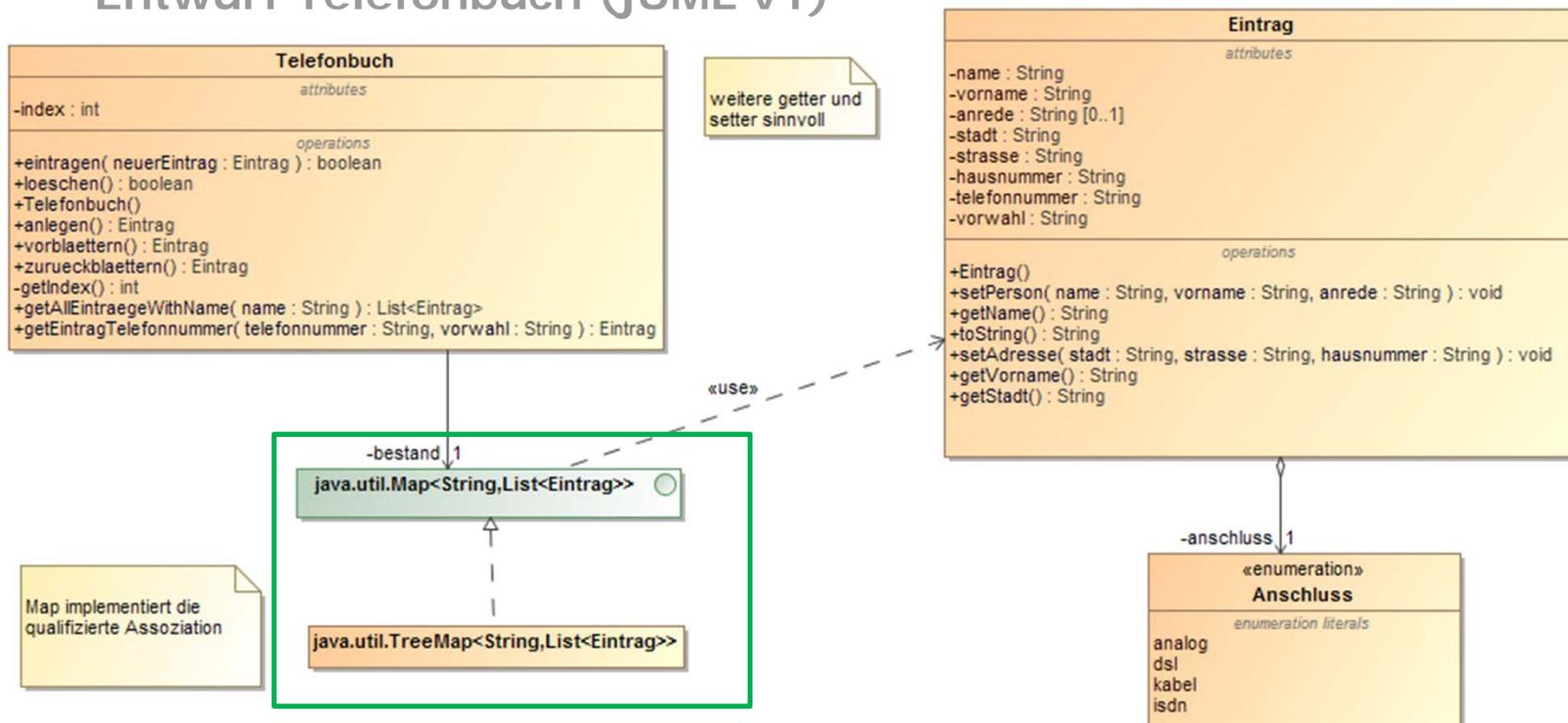
Entwurf Telefonbuch (jUML v0)



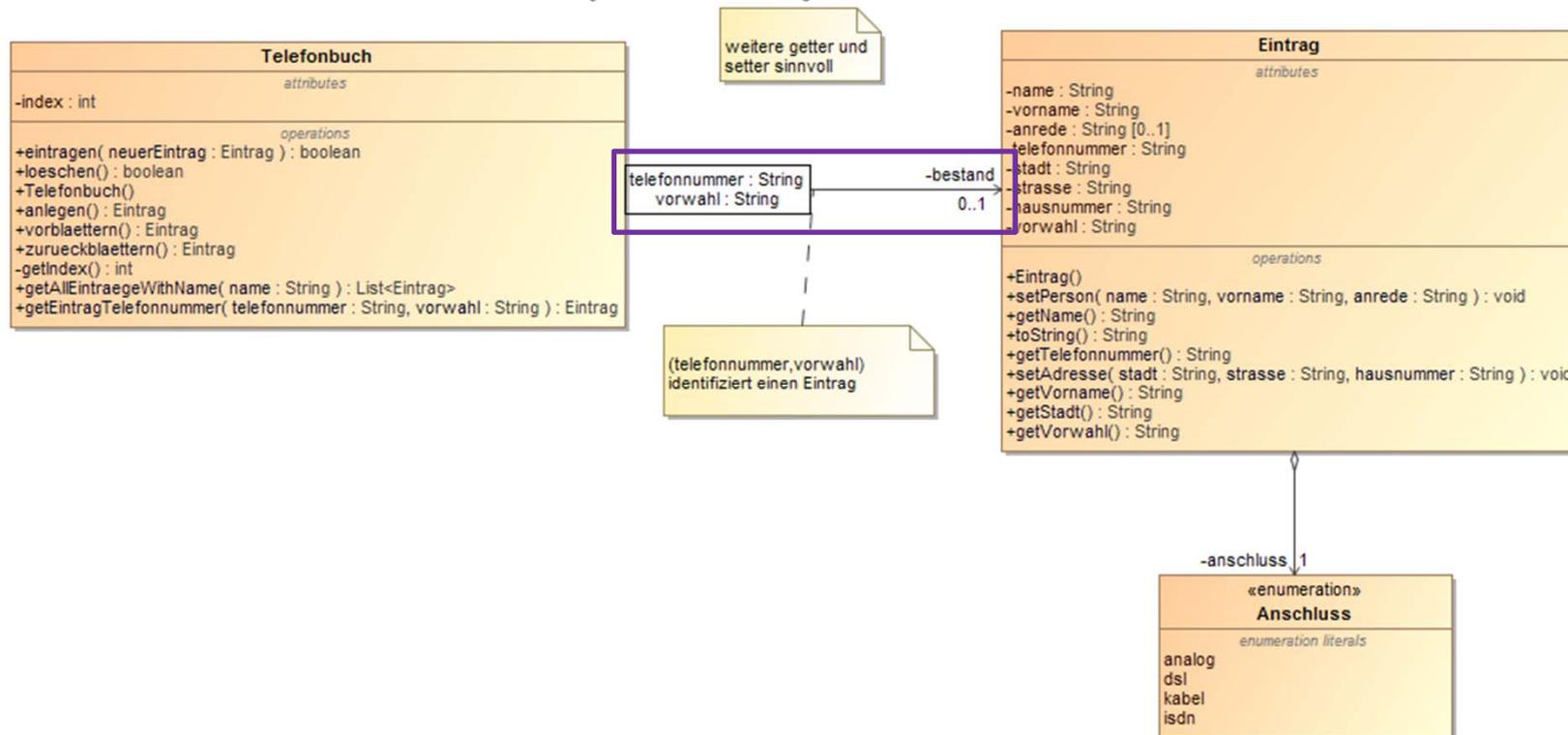
Entwurf Telefonbuch (dUML v1)



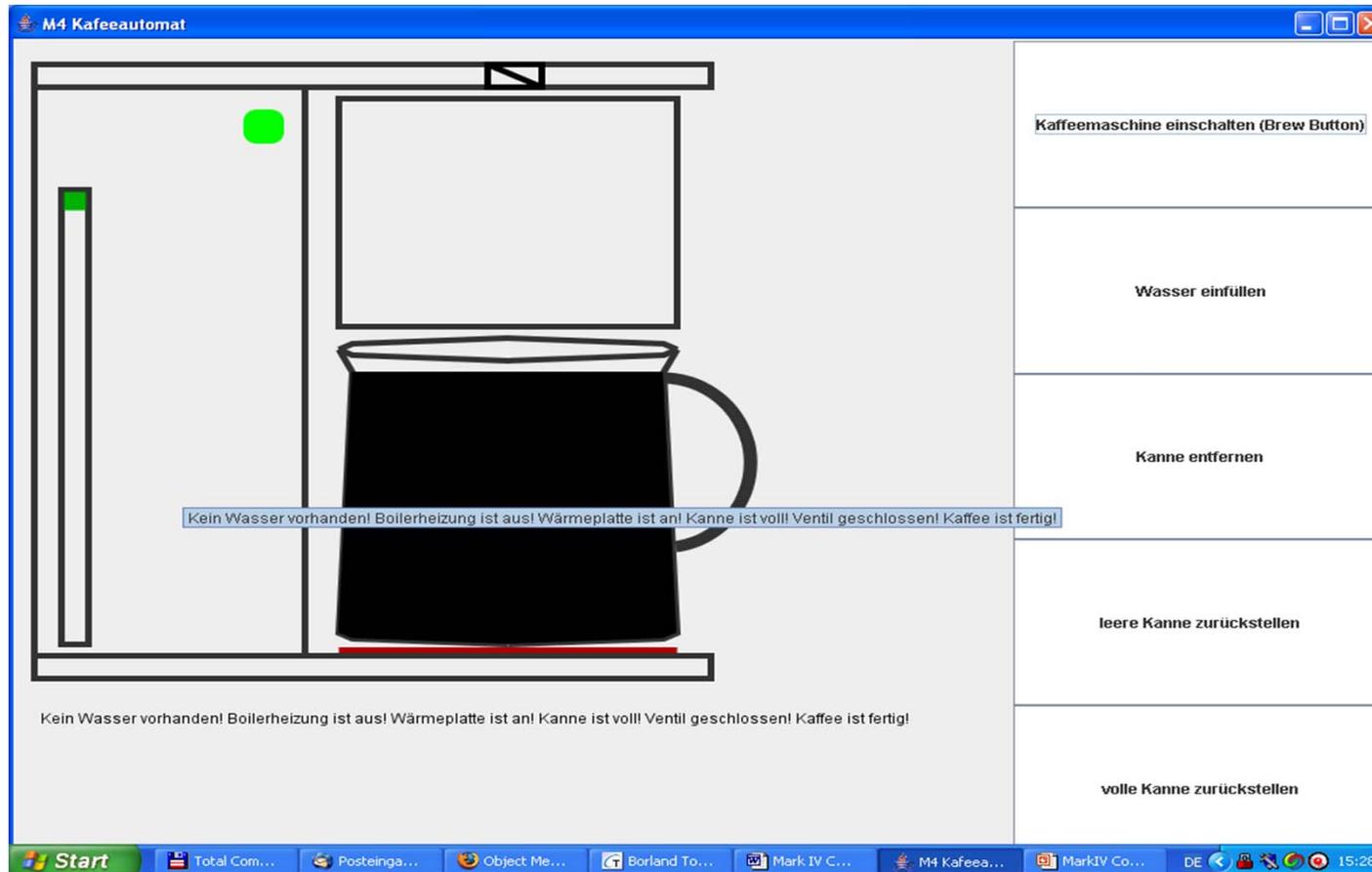
Entwurf Telefonbuch (jUML v1)



Entwurf Telefonbuch (dUML v2)



FALLSTUDIE MARK IV COFFEEMAKER /1/,/2/



Teile und Funktionen von MarkIV Coffeemaker

Heizung für den Boiler (**boilerOn/boilerOff**)

Heizung für die Wärmeplatte (**warmerOn/warmerOff**)

Sensor für die Wärmeplatte (**warmerEmpty, potEmpty, potNotEmpty**)

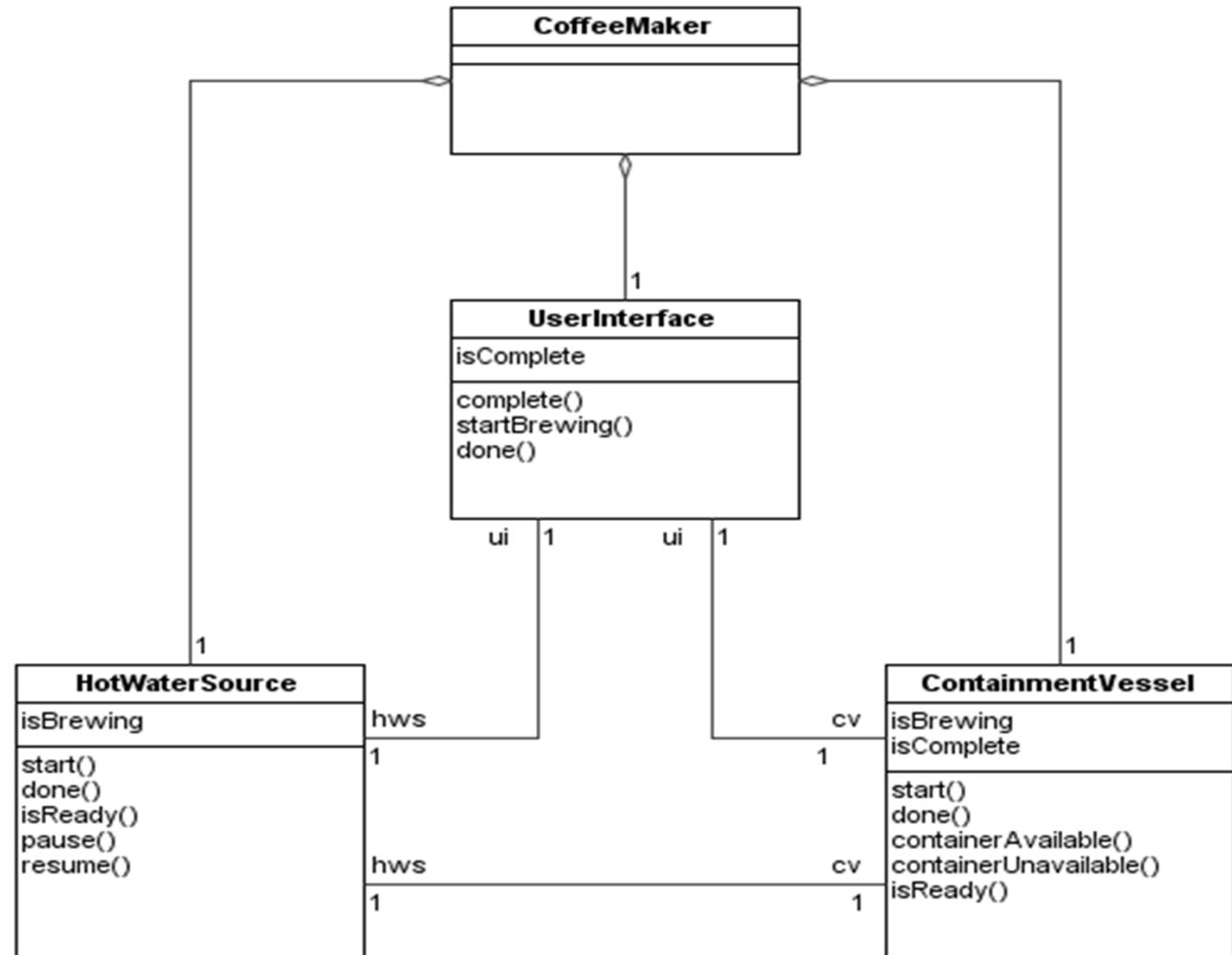
Sensor für den Boiler (**boilerEmpty, boilerNotEmpty**)

Zubereitungsknopf (**brew**)

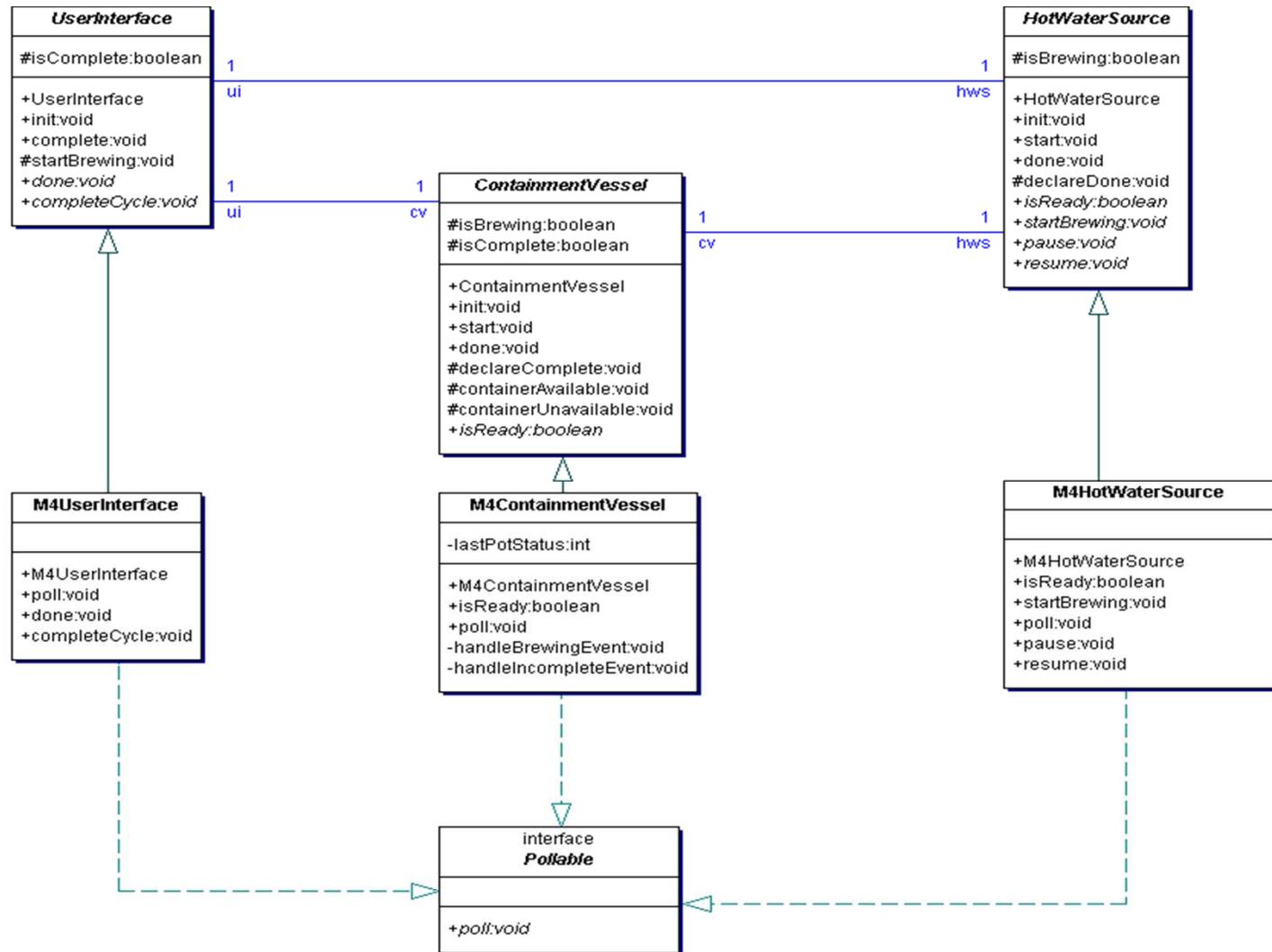
Anzeigelampe (**indicatorOn/indicatorOff**) leuchtet auf, wenn der Zubereitungsvorgang beendet und der Kaffee fertig ist.

Druckventil (**valveOpen/valveClosed**), das geöffnet wird, um den Druck im Boiler zu reduzieren; der Druckabfall unterbricht den Wasserzufluss zum Filter; das Ventil kann geöffnet oder geschlossen sein.

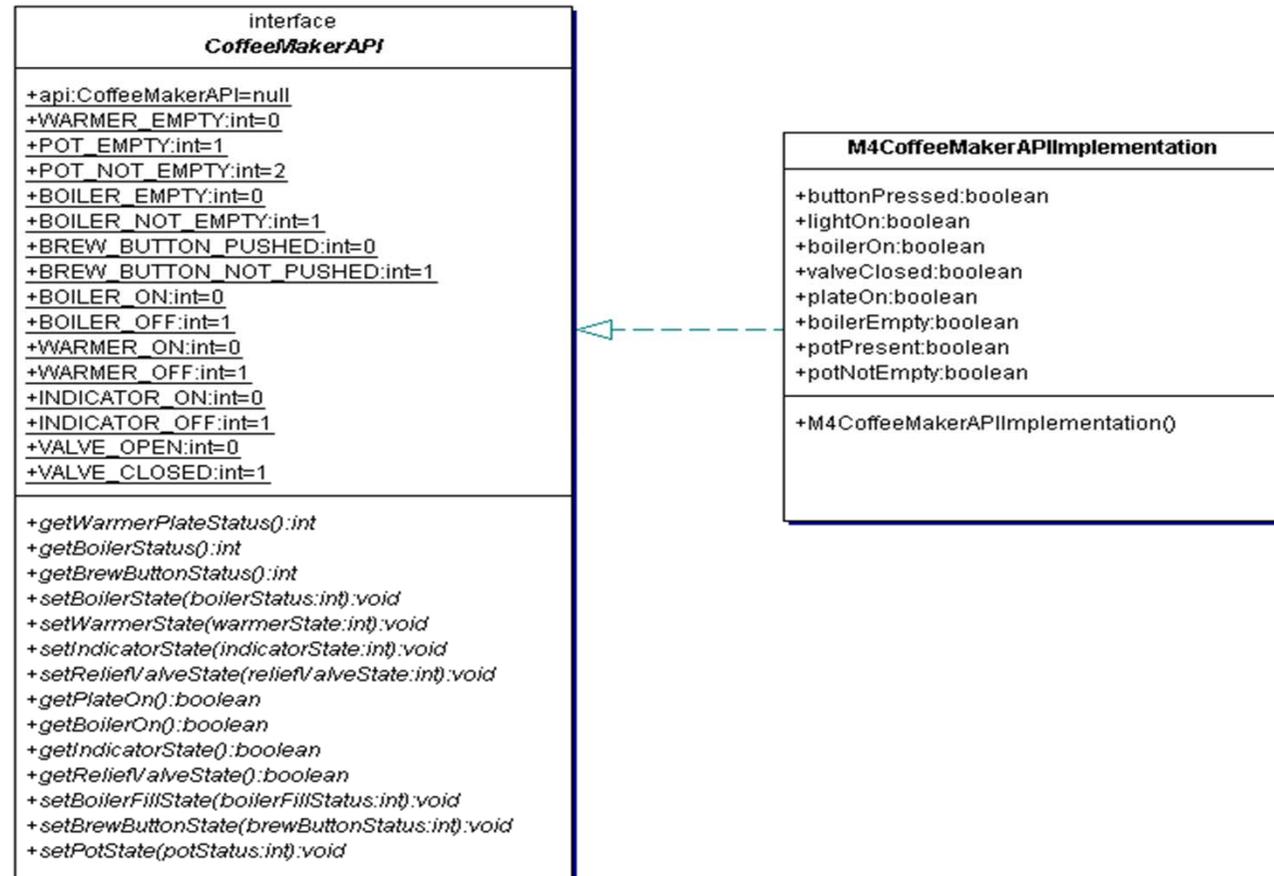
OOA: Klassen- diagramm



OOD



CoffeeMaker API Spezifikation der Hardware- funktionen



Schnittstellen und abstrakte Klassen in Java

Abstrakte Klasse	Schnittstelle (Interface)
Attribute, Konstanten, Operationen	Operationen und ggfs. Konstanten
Kann Default-Verhalten festlegen	Kann KEIN Default-verhalten festlegen
Default-Verhalten kann in Unterklassen überschrieben werden	Überschreiben von Methoden ist nicht möglich
Unterklasse kann nur von einer Klasse erben	Eine Klasse kann mehrere Schnittstellen implementieren
Verwendung für Implementierungsvererbung	Verwendung für Spezifikationsvererbung

MarkIV CoffeeMaker Entwurfsüberlegungen

```
public class CoffeeMaker {  
    public static void main(String[] args) {  
        CoffeeMakerAPI api = new  
M4CoffeeMakerAPIImplementation();  
        M4UserInterface ui = new  
M4UserInterface(api);  
        M4HotWaterSource hws = new  
M4HotWaterSource(api);  
        M4ContainmentVessel cv = new  
M4ContainmentVessel(api);
```

```
        ui.init(hws,cv);  
        hws.init(ui,cv);  
        cv.init(ui,hws);  
        while(true) {  
            ui.poll();  
            hws.poll();  
            cv.poll();  
        }  
    }  
}
```

Implementierung

Implementierung der Interfaces/Klassen des OOD-Klassendiagramms

Wie wird nun die Kaffeemaschine MarkIV zum ablauffähigen Programm?

- Klasse CoffeeMaker als Kommandozeilenprogramm (aber da sieht man nichts, Endloszyklus)
- GUI zur Simulation der Kaffeemaschine, d.h. etwas zum Spielen 😊
- Erweiterung unserer Klasse CoffeeMaker um eine GUI
- Implementierung mit dem Observer Pattern
- Nächster Schritt: MarkIV mit Threads für die einzelnen Sensoren

Referenzen

- (1) Robert Martin: Designing Object-Oriented C++ Applications Using the Booch Method. Prentice Hall, 1995
- (2) Robert Martin: UML for Java Programmers. Prentice Hall, 2003
- (3) Birgit Demuth (Hrsg.): Softwaretechnologie für Einsteiger. Pearson Studium, 3. geänderte Auflage, 2019

Weitere Quellen zur Vorbereitung auf die Klausur

- Vorlesungsfolien und Übungsmaterial
- Learning Outcomes
- INLOOP-Aufgaben
- Auditorium
- Lernraum
- Klausuren auf IFSR-Server (<ftp://ifsr.de/klausuren/SWT/>)

Ende