3. Komponenten, Entwurfsmuster, Rahmen

3.1 Komponenten

Objektorientierter Entwurf: Gliederungseinheit ist die Klasse

- Klasse besteht aus
  - Schnittstelle (Klassendefinition)
  - Implementierung
- Schnittstelle und Implementierung bilden eine Einheit

Wünsche:

- Gliederungseinheiten oberhalb von Klassen
- Trennung von Schnittstelle und Implementierung
- ⇨ **Komponenten**

Komponente – Zusammenfassung mehrerer Klassen zu einer logischen Einheit

- Elemente der Komponente lösen zusammen ein Teilproblem
- Komponente wird typisch als Ganzes verwendet/wiederverwendet
- Bietet die Möglichkeit, nach außen nur eine (gemeinsame) Schnittstelle anzubieten

(Selbständige) Schnittstelle – Komponente, die nur aus einer Leistungsbeschreibung ohne jegliche Implementierung besteht

- keine Klasse
- Leistungen können je nach Umfang der Schnittstelle durch Klassen oder Komponenten erbracht werden.
- Leistungsbeschreibung und Leistungserbringung sind völlig entkoppelt
- Jede Implementierung, welche alle in der Schnittstelle beschriebenen Leistungen erbringt, ist zulässig

Beispiel:

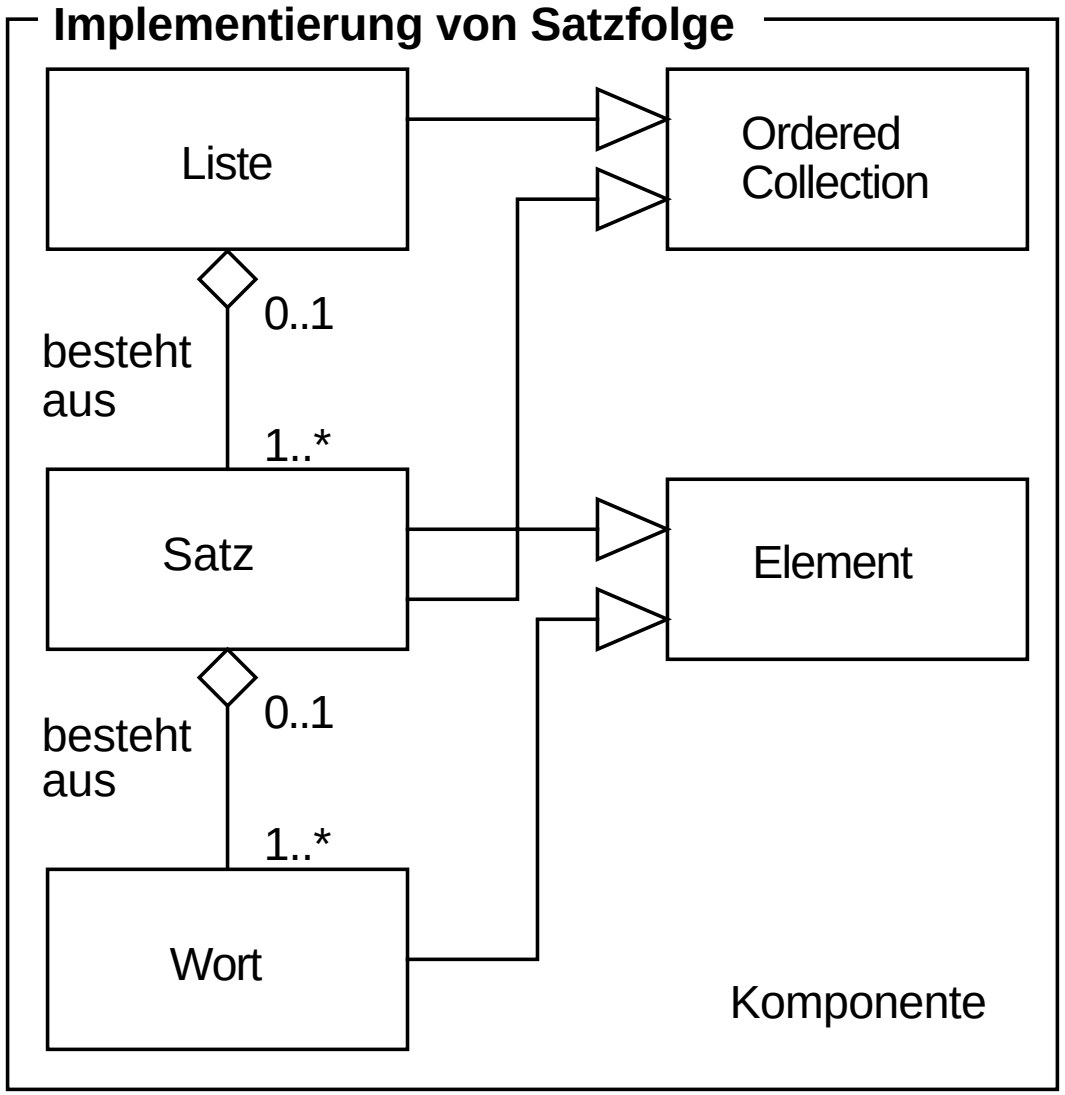
Satzfolge: Schnittstelle

listeErzeugen
satzBilden
wortErzeugen
wortAufPositionEinfügen

satzLöschen
wortAusPositionLöschen

ersterSatz
nächster Satz

erstesWort
nächstesWort
wortSuchen
wortErneutSuchen



3.2 Entwurfsmuster

3.2.1 Grundsätzliches

Motivation:

- Bereitstellung bewährter, vorgefertigter Lösungen für wiederkehrende Entwurfsprobleme
- Schaffung einer begrifflichen Basis und Terminologie für die Kommunikation über solche Probleme

Entwurfsmuster (design pattern) – spezielle Komponente, die eine allgemeine, parametrierbare Lösung für ein typisches Entwurfsproblem bereitstellt.

Grundsätzlich gibt es eine Fülle von Mustern, sowohl allgemeiner Art als auch für spezifische Anwendungsbereiche.

Musterkataloge erschließen das Wissen über Muster und machen die Muster wiederverwendbar.

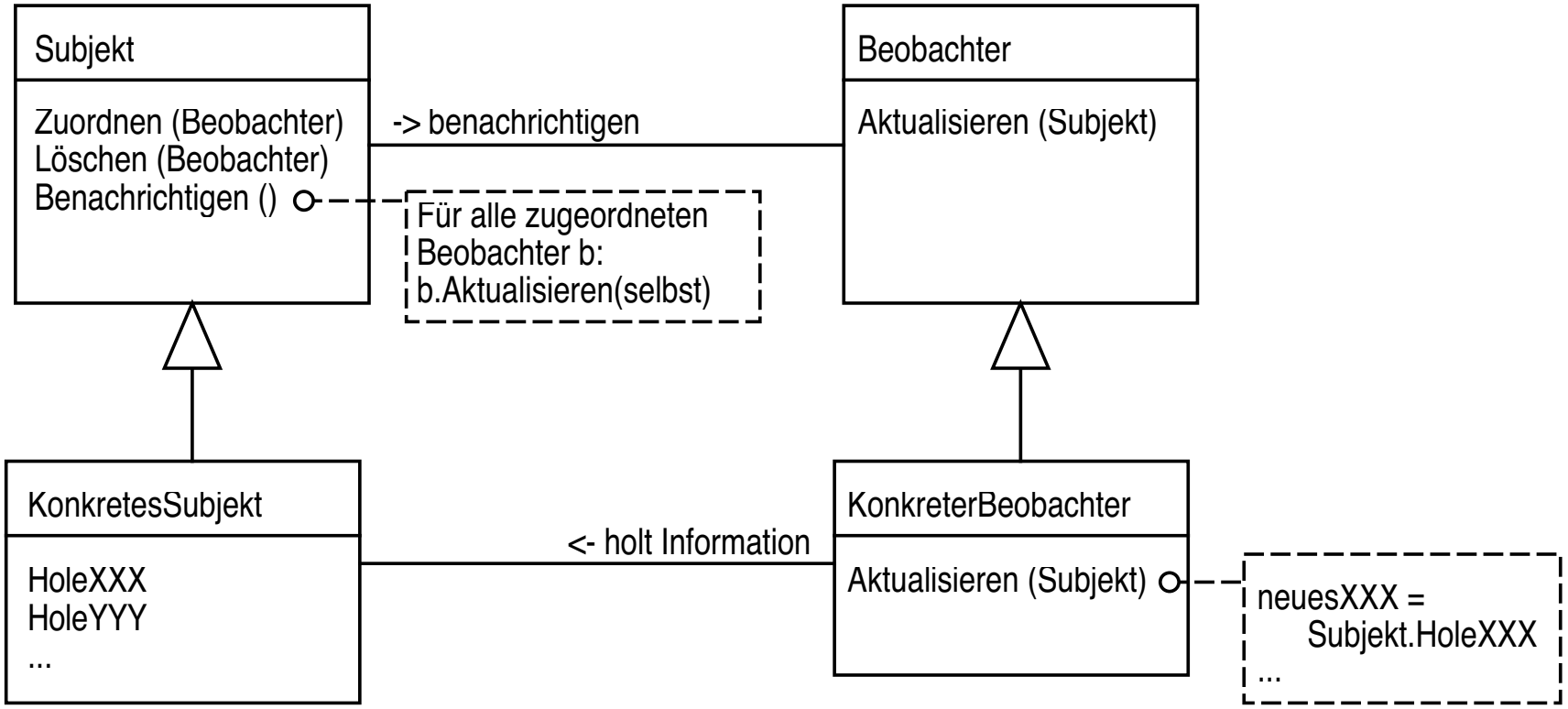
3.2.2 Beispiel: Das Beobachter-Muster (observer pattern)

Intention: Objekte können Daten bei einem Informationsanbieter abonnieren. Bei jeder Änderung der subskribierten Daten werden die Abonnenten automatisch über die Änderung informiert bzw. werden ihnen die geänderten Daten zugestellt.

Typische Anwendungen:

- Entkopplung voneinander abhängiger Objekte durch Ersetzung direkter Kommunikation über Aufrufe durch eine indirekte über Benachrichtigung
- Trennung von Informationsbereitsteller und einer Menge von Informationsverarbeitern bzw. Darstellern

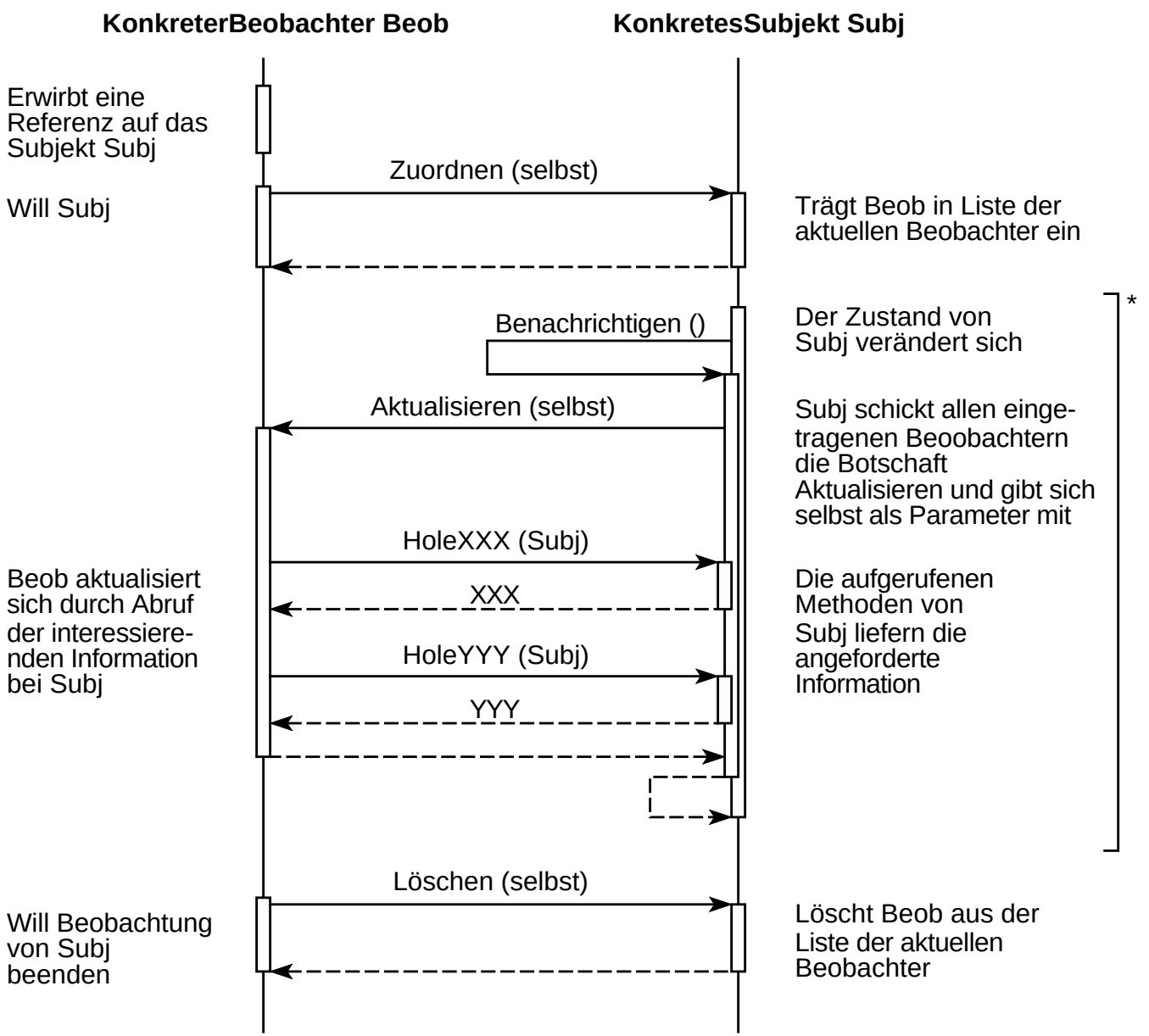
Struktur des Musters:



Arbeitsweise:

- Jedes Objekt der Klasse Subjekt führt eine Liste von Beobachtern, welche an Veränderungen im Zustand dieses Objekts interessiert sind. Zuordnen bzw. Löschen fügt Beobachter in die Liste ein bzw. entfernt sie.
- Nach jeder Veränderung schickt das Gegenstandsobjekt sich selbst die Botschaft Benachrichtigen. Diese iteriert die Liste der Beobachter und schickt jedem Beobachter die Botschaft Aktualisieren. Das benachrichtigende Objekt wird als Parameter mitgegeben. Ggf. können weitere Informationen (z.B. die Art des eingetretenen Ereignisses) als Parameter mitgegeben werden.
- Jeder benachrichtigte Beobachter reagiert, indem er beim benachrichtigenden Gegendstandsobjekt mit HoleXXX-Botschaften die ihn interessierenden Informationen abruft.
- Alternativ können der Update-Botschaft die veränderten Daten gleich mitgegeben werden, wodurch der Abruf durch HoleXXX entfällt. Dieses Bringprinzip ist effizient, koppelt aber Gegenstand und Beobachter stärker als das mit HoleXXX realisierte Holprinzip.

Benutzungs-szenario des Musters:



* Dieser Teil kann mehrfach iteriert werden

3.2.3 Entwurf mit Mustern

- Voraussetzung: Grundschatz an Mustern kennen (Lernen / Erfahrung / Entwürfe und Code anderer lesen bzw. inspizieren)
- Bei der Modularisierung Anwendungssituationen für Muster erkennen ➪ entsprechende Muster verwenden
- Allgemeine Muster wie Beobachter, Mediator oder Fabrik beschreiben weitgehend reine Lösungsstrukturen und haben keine Entsprechung in Objekten der Aufgabenstellung
- Daneben gibt es auch anwendungsspezifische Muster, welche die Bildung von Aufgabenmodellen erleichtern

3.3 Rahmen

Rahmen (framework) – Eine Menge kooperierender Module, die das Grundgerüst für die Lösung einer bestimmten Klasse von Problemen bilden.

- Konkrete Lösungen entstehen durch Ergänzung/Spezialisierung des Rahmens durch problemspezifische Module.
- Werden Rahmen verwendet, so bestimmen diese weitestgehend die Grundarchitektur der Lösung.
- Der Umfang eines Rahmens kann stark variieren:
  - Eng: Lösung eines Teilproblems, z.B. Realisierung einer Benutzerschnittstelle, Verwaltung von Dateien mit Vergabe und Prüfung von Zugriffsrechten oder die Grundfunktionalität eines Editors
  - Umfassend: Lösungsgerüst für ein vollständiges System, z.B. Rahmen für das Schaltergeschäft in einer Bank.
- Manchmal werden in einer Architektur mehrere Rahmen gleichzeitig verwendet. ➪ Sehr anspruchsvoll: Unterschiedliche Architekturprinzipien der verschiedenen Rahmen müssen in gemeinsame, kohärente Gesamtarchitektur integriert werden.