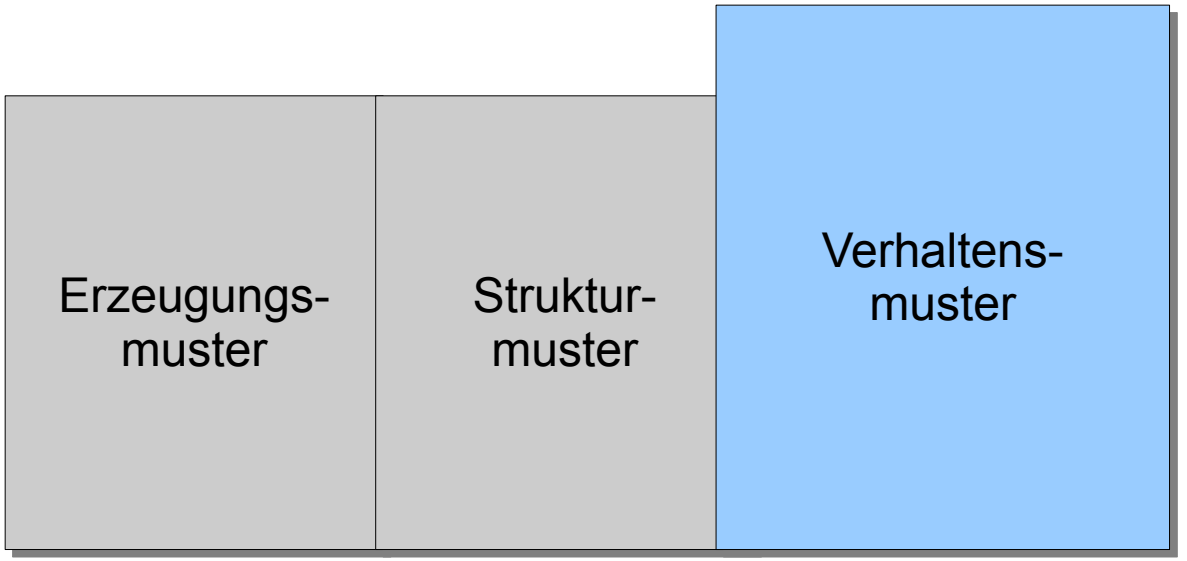
Entwurfsmuster
Der Beobachter

Don't call us, we will call you
(Hollywood)

Beobachter

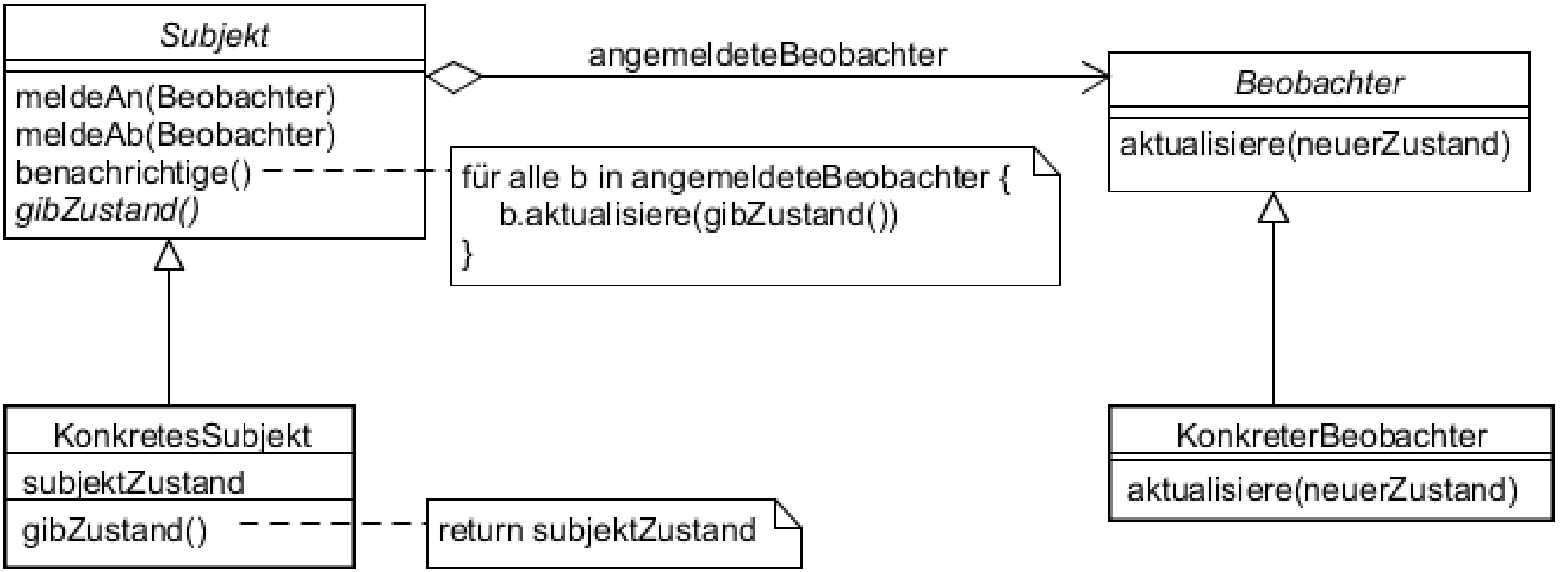
- Klassifikation
  - Objektbasiertes Verhaltensmuster
  - Langlebig
- Alternativname: Observer, Listener



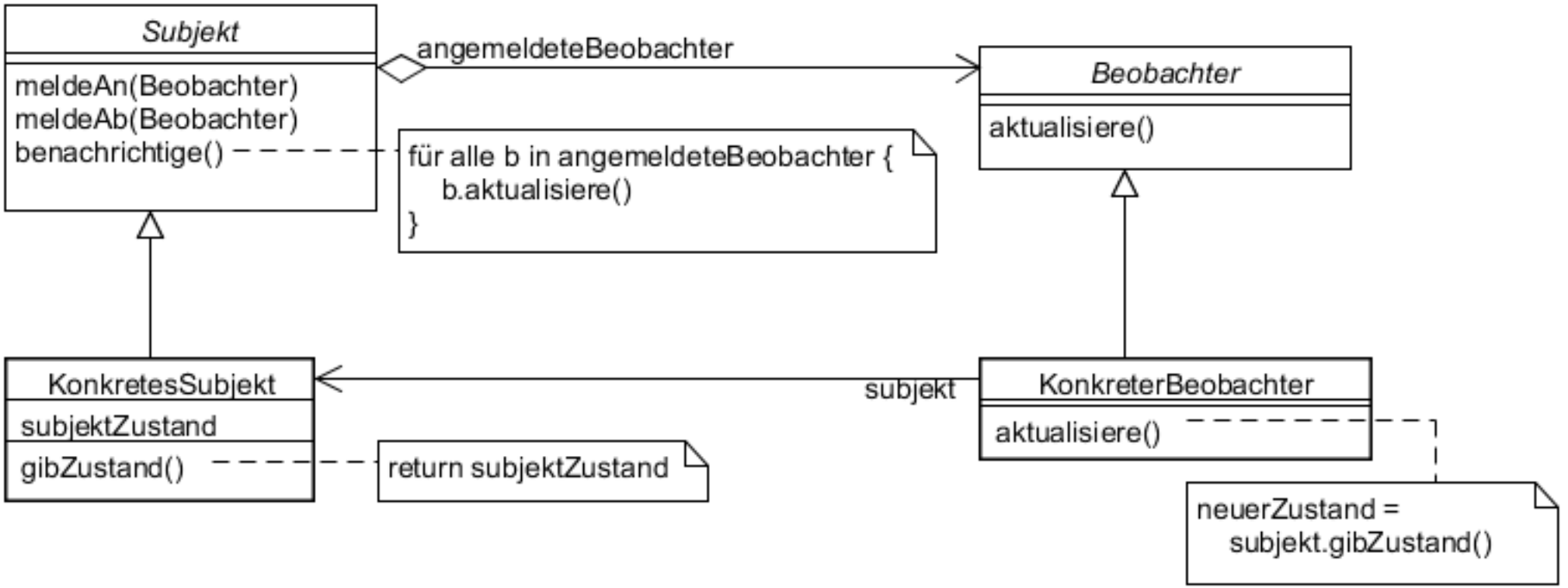
Motivation

- Änderungen an einem Objekt einer Anzahl anderer Objekte mitzuteilen
  - 1-zu-N Beziehung („eins-zu-vielen“)
  - Broadcast-Kommunikation (genauer: Multicast)
- Möglichst geringe Kopplung der Objekte
  - Variabel zur Laufzeit des Systems
- Kommunikation nur im Änderungsfall
  - Statt periodische Anfragen („polling“)

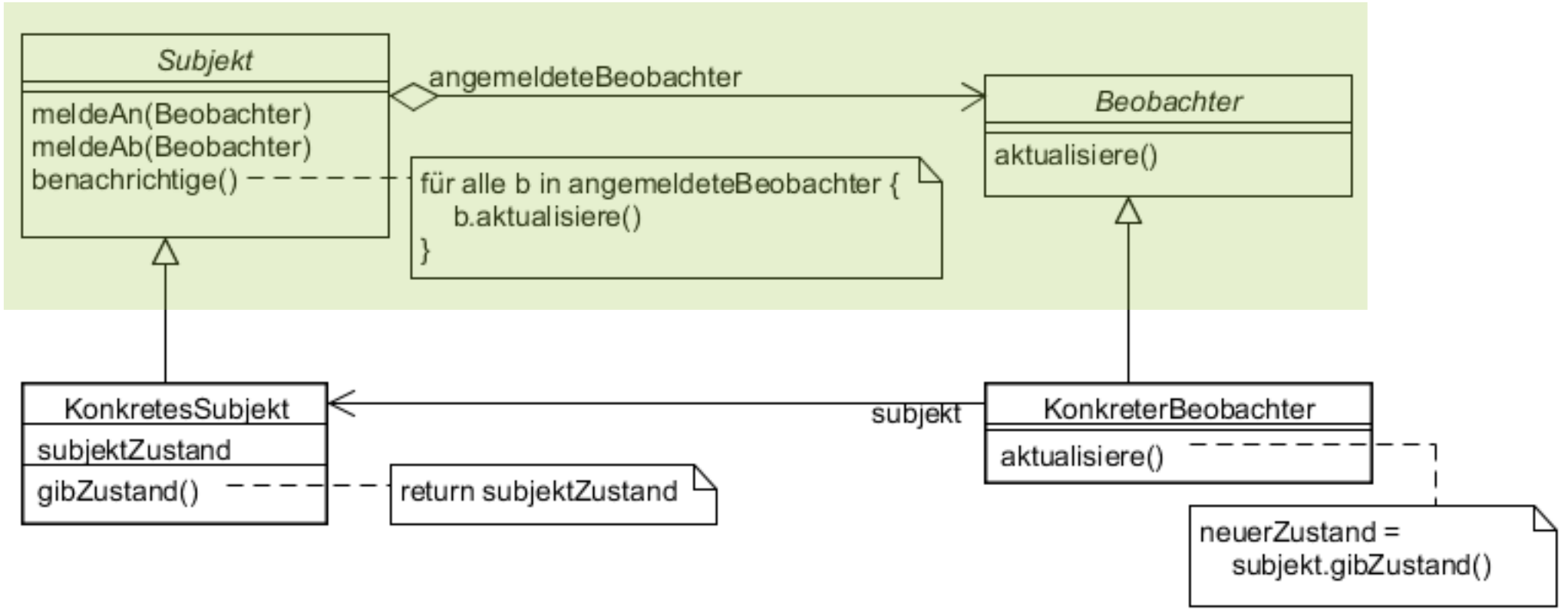
UML-Diagramm (abstrakt)



UML-Diagramm (abstrakt)



UML-Diagramm



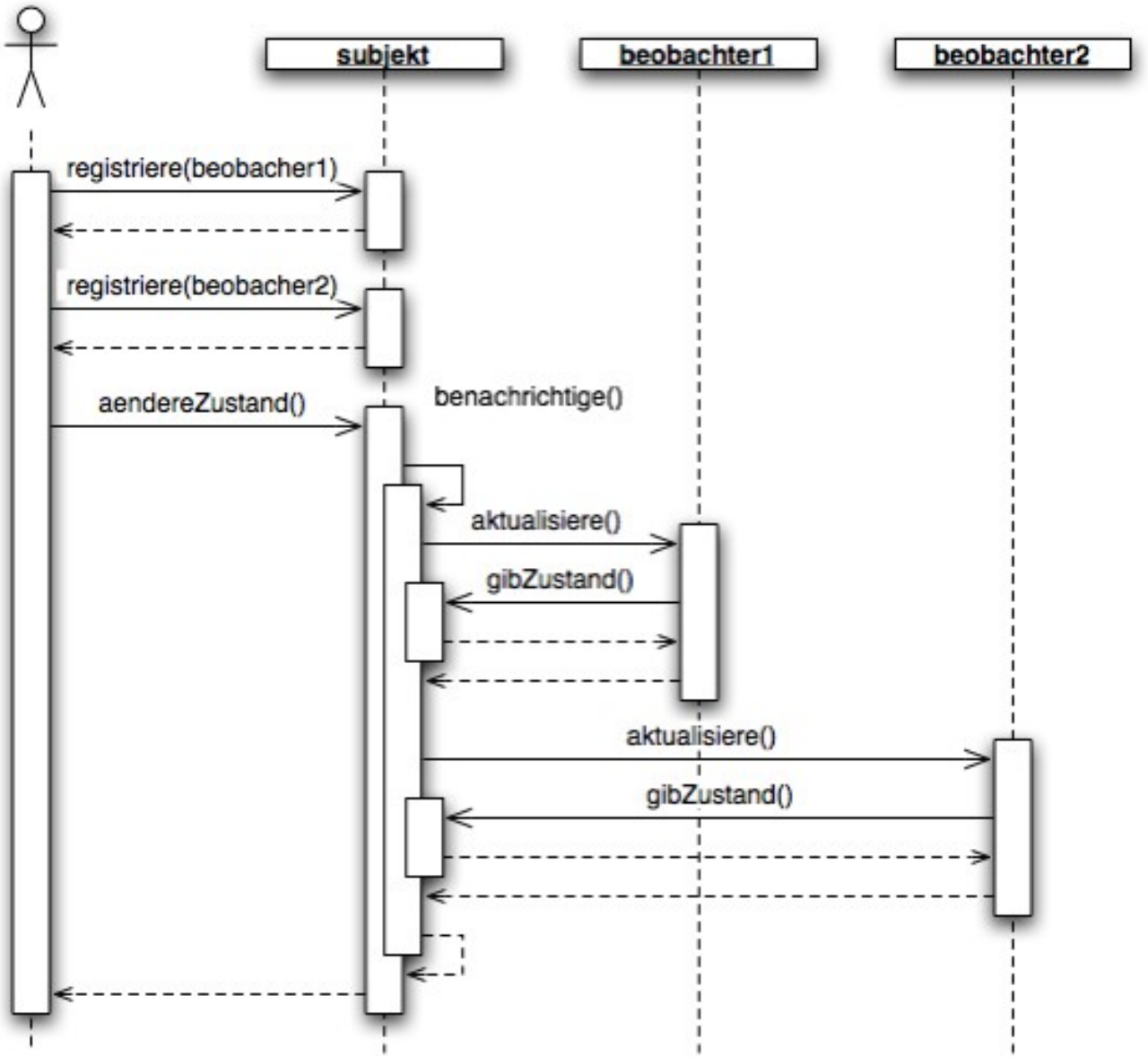
Anwendung

- Der Beobachter hilft besonders, wenn
  - ein Objekt seinen Zustand ändert
    - Weshalb sich andere Objekte ebenfalls ändern sollen
    - Deren Anzahl und Typ erst zur Laufzeit bekannt sind
    - Und die dies nur zeitweise tun sollen
- Der Beobachter entkoppelt das Subjekt von
  - Der Anzahl der Beobachter-Objekte
  - Den konkreten Typen der Beobachter-Objekte
  - Festen Referenzen auf die Beobachter-Objekte

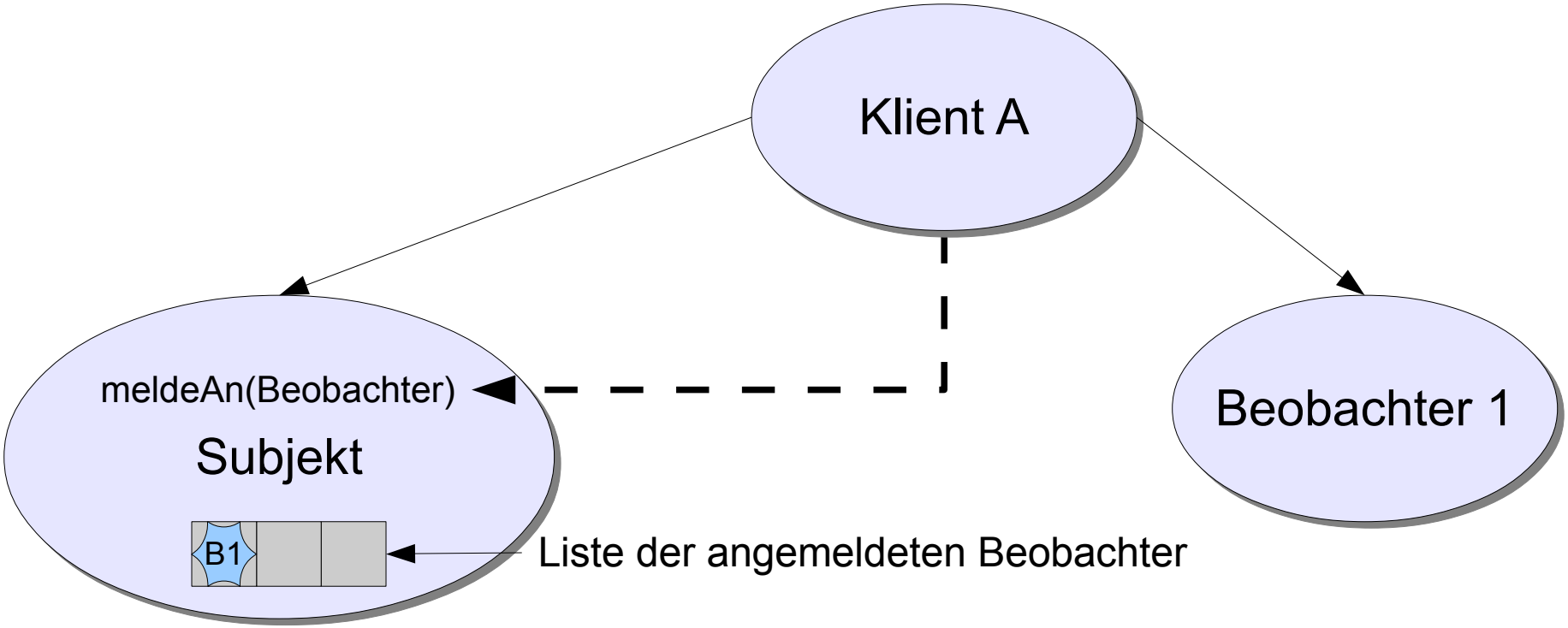
Interaktion

- Zustandsänderung eines beobachteten Attributs
  - Subjekt benachrichtigt angemeldete Beobachter
  - Beobachter befragen Subjekt nach relevanten Informationen
    - Alternative: Subjekt liefert alle Informationen in der Benachrichtigung
  - Beobachter bringt seinen Zustand in Einklang mit Subjekt

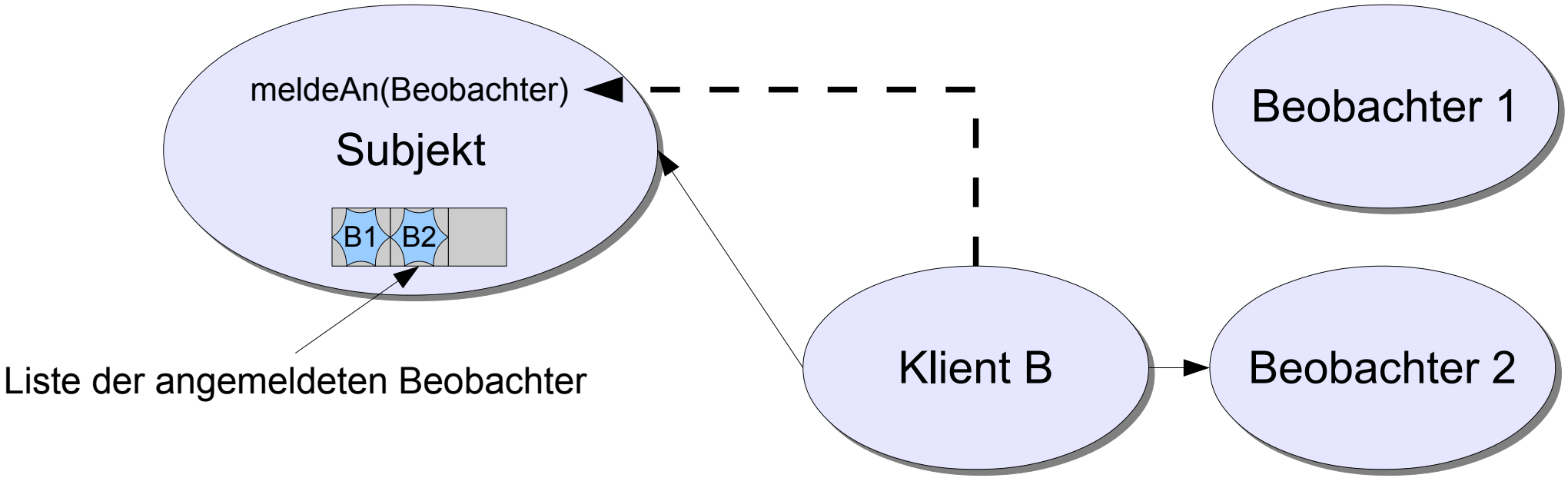
Interaktion graphisch



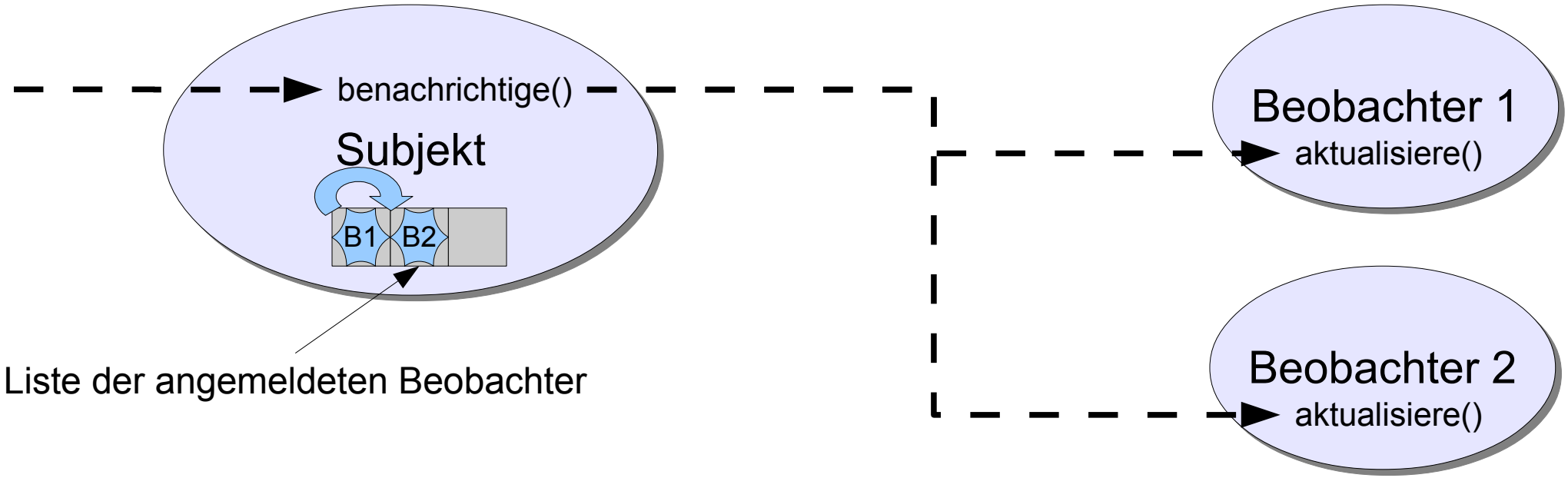
Interaktion



Interaktion

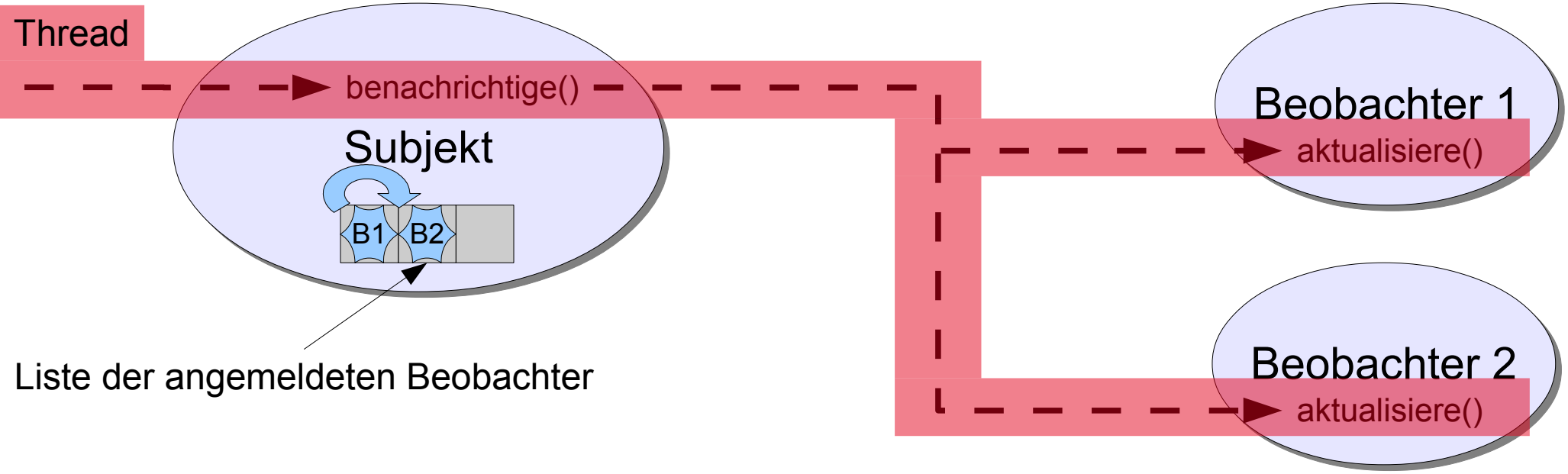


Interaktion



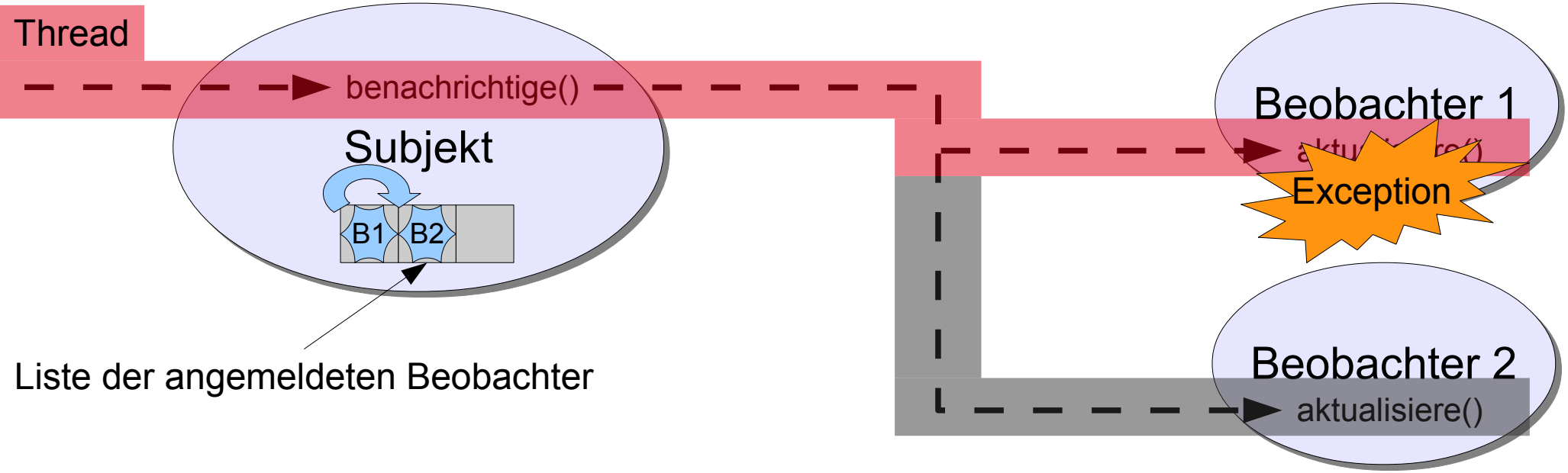
Interaktion: Erkenntnisse

- Aufrufender Thread „besucht“ die Beobachter
  - Aufrufe seriell, könnte Verzögerung bedeuten
  - Benötigt eventuell Thread-Management im Subjekt



Interaktion: Erkenntnisse

- Ungefangene Exception in einem Beobachter
  - Stoppt die Benachrichtigung und tötet den Thread
  - Benötigt eventuell Exception-Handling im Subjekt



Zusammenfassung: Vorteile

- Lose Kopplung zwischen Subjekt und Beobachtern
  - Subjekt besitzt nur Liste mit Objekten vom abstrakten Typ
  - Beobachter kennen das Subjekt meistens nicht
- Polling-freie Broadcast-Benachrichtigung bei Änderungen
  - Dynamisch (An- und Abmeldung)

Zusammenfassung: Nachteile

- Bei ungünstiger Anmeldung
  - Benachrichtigungs-Kaskade
  - Endlosschleife durch zirkuläre Abhängigkeiten
- Threads „wandern“ eventuell tief in Fremdcode
  - Zeitliches Verhalten nicht vorhersagbar
- Ungefangene Exceptions haben Auswirkungen
  - Exception-Behandlung im Subjekt meist nur „generisch“ möglich (kein Kontext mehr)

Implementierungen

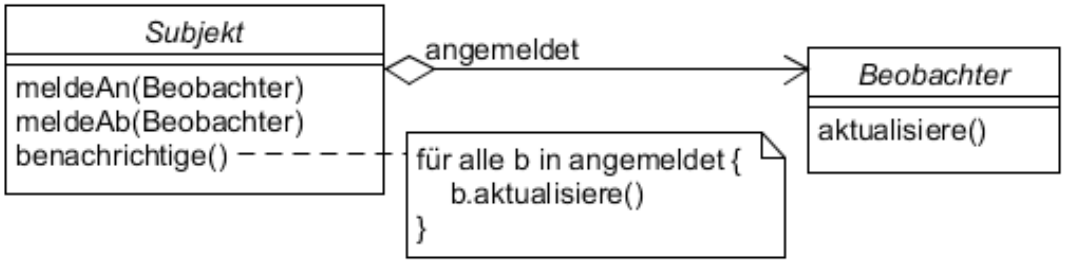
- Java
  - Listener-Architektur
    - ActionListener, MouseListener
- .NET
  - Event-/Delegate-Mechanismus
    - EventHandler (C#), AddHandler (VB)
- In vielen Sprachen native Beobachter-Features
- Oft implizit ausgeprägt
  - Interrupts, Callbacks, Hooks

Ausprägungen

- Push-Modell
  - Jede Benachrichtigung enthält alle Informationen
  - aktualisiere()-Methode mit Parametern
  - Spezifische Schnittstelle
- Pull-Modell
  - Nur Benachrichtigung, keine zusätzlichen Informationen
  - Beobachter müssen Informationen selbst einholen
    - Eventuell notwendig: Referenz von Beobachter auf Subjekt

Zusammenfassung

- Beobachter, Listener
- Objektbasiertes Verhaltensmuster



- Dynamisch abonnierte Broadcast-Änderungsbenachrichtigung
  - Lose Kopplung der Kommunikationsteilnehmer
- Verschiedene Ausprägungen
- Problemstellen
  - Threading
  - Exceptions