Der Klassenentwurf

Entwurfsprinzipien:

- Sorge für wenig bzw. lose Kopplung der einzelnen Klassen
  (z.B. durch Verwendung von Schnittstellen)
- hoher Grad an Kohäsion
  (d.h. jede Methode soll nur eine Aufgabe erfüllen, jede Klasse eine Einheit darstellen)
- Vermeidung von Code-Duplizierung
- Datenkapselung

Einschub: Schnittstellen (interfaces)

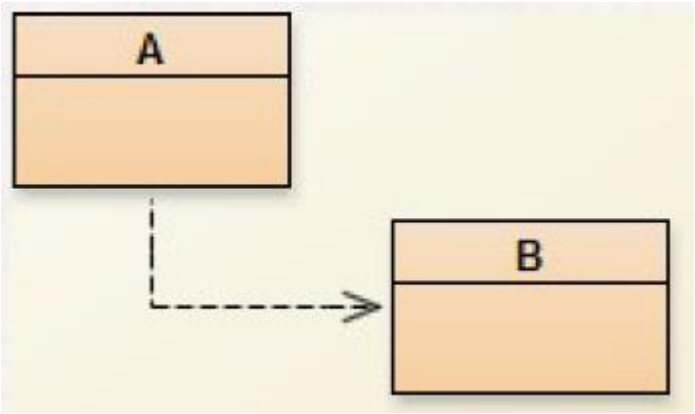
Ein Interface in Java ist eine Spezifikation eines Typs in Form eines Typnamens und einer Menge von Methoden ohne Rumpf.
Eine "Unterklasse" eines Interface muss diese Methoden implementieren.

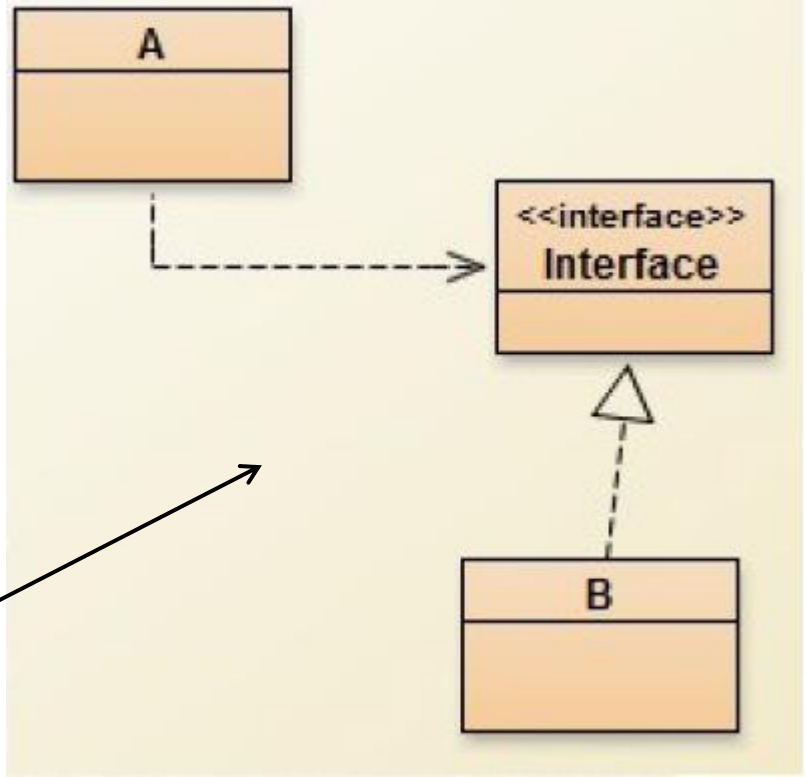
```
public interface MachDas {
   public void lobMich();
}

public class IchTus implements MachDas {

    public void lobMich() {
       System.out.println("Du bist spitze!");
    }
}
```

- Erben von einer Superklasse und Implementieren mehrerer Interfaces ist möglich:

  public class IchTus extends Superklasse implements Interface1, Interface2 {…}

- Entkopplung durch Interfaces





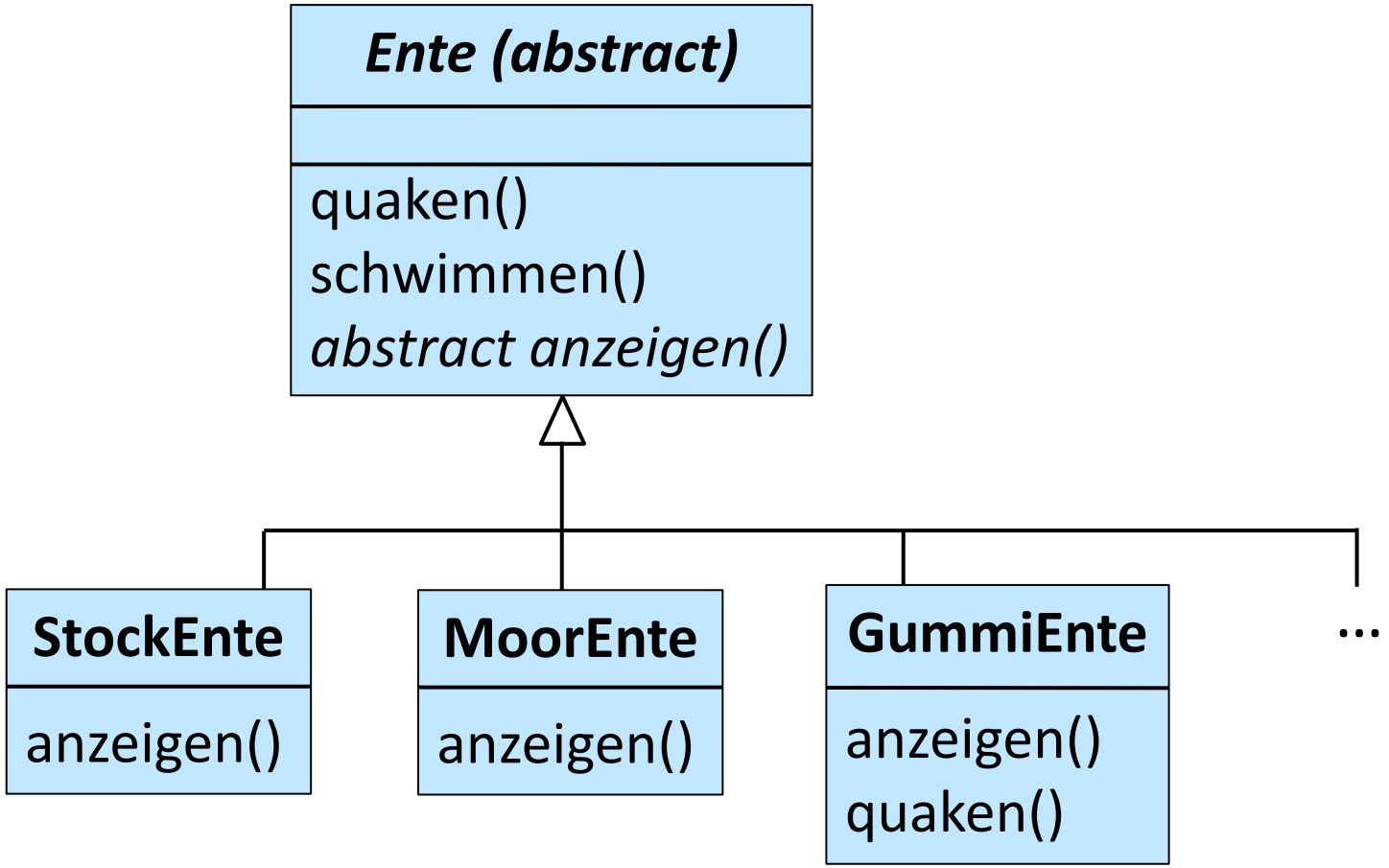
Klassen A und B sind entkoppelt:
Änderung von B betrifft A nicht

Strategiemuster

Es gibt mehrere konkrete Strategien, die ein Strategie-Interface implementieren und eine Kontextklasse, die das Interface als Attribut nutzt, um dann die Strategien je nach Bedarf wechseln zu können.

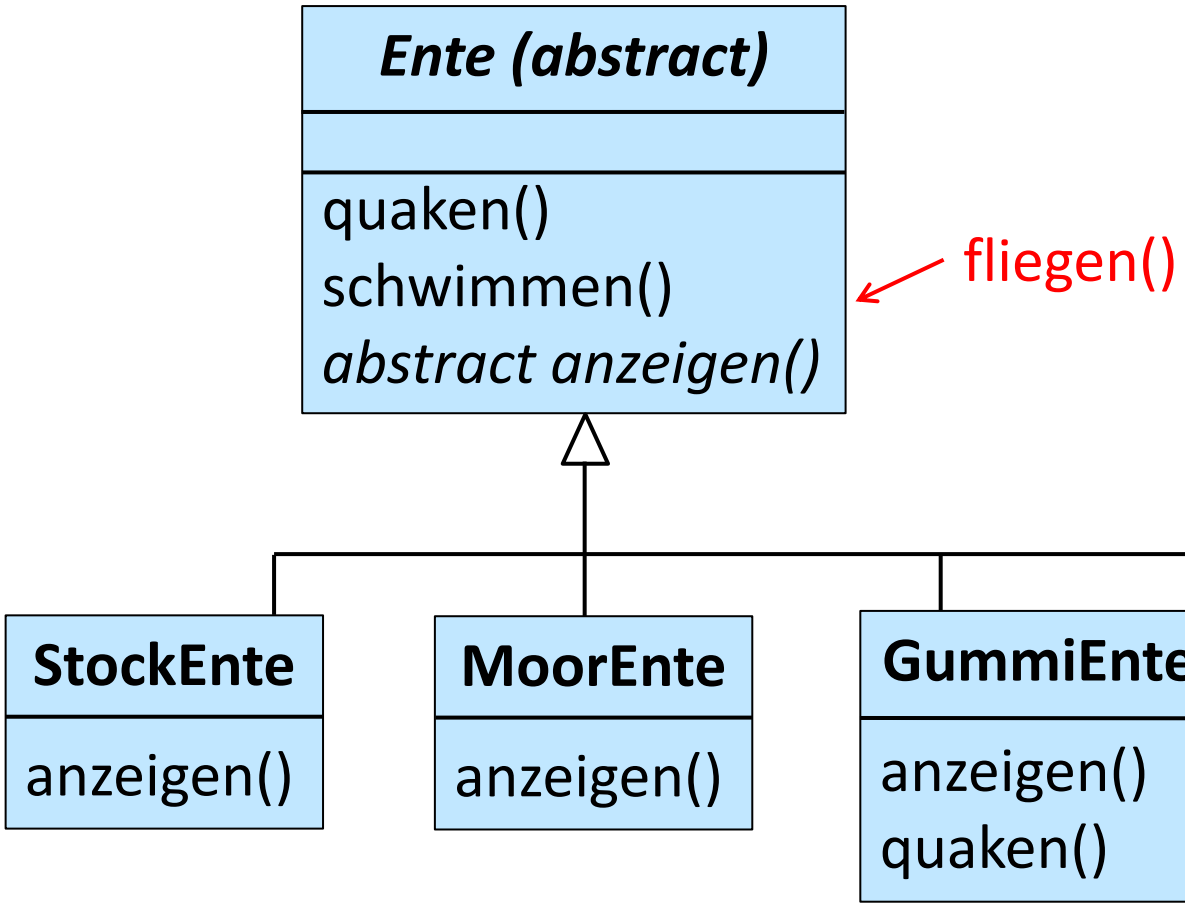
Beispiel:

Es sollen verschiedene Entenarten modelliert werden, die quaken und schwimmen können….



…und jetzt noch fliegen!

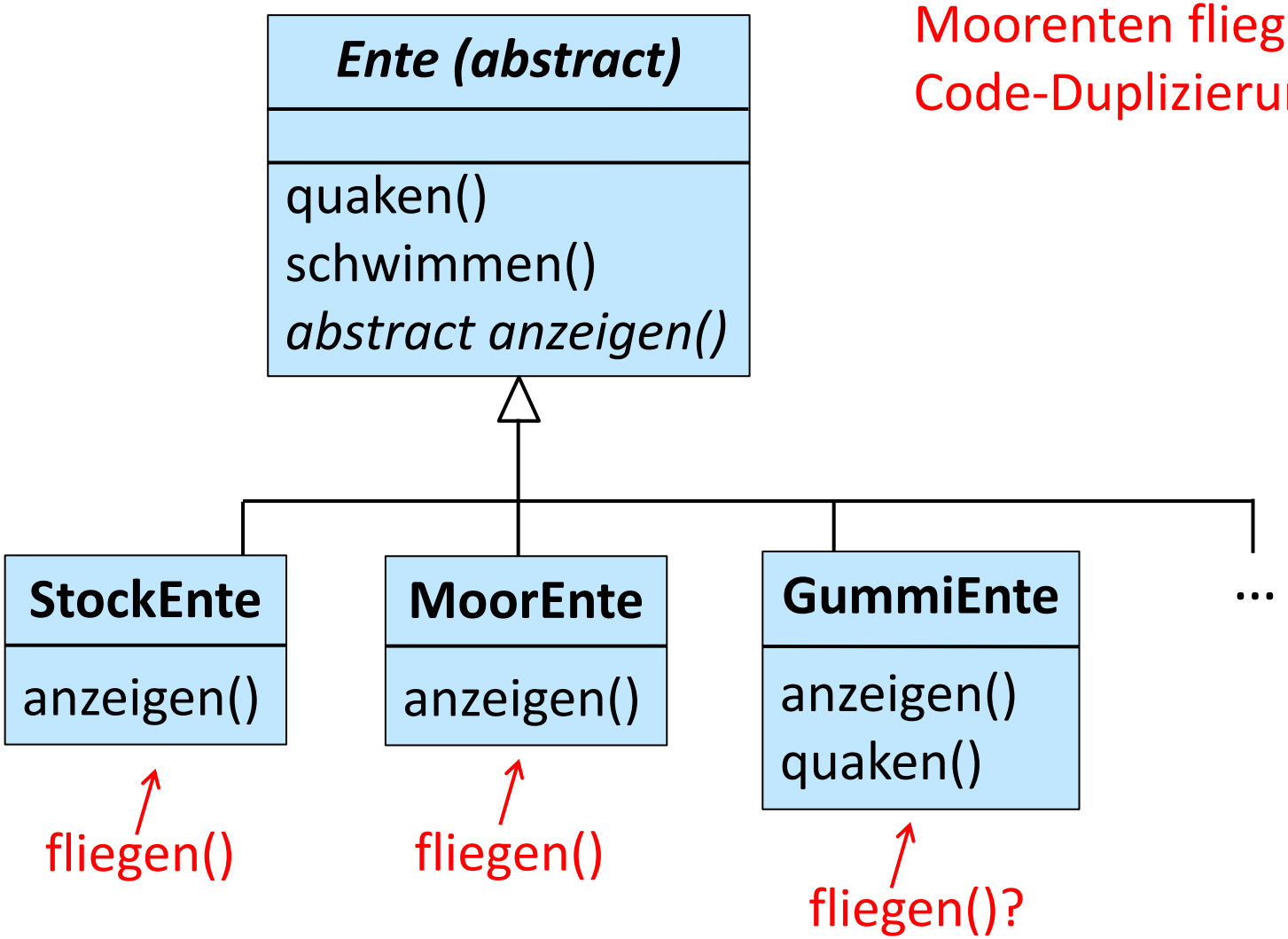
Wohin mit "fliegen"?



Aber Gummienten können nicht fliegen!
Andere Arten vielleicht auch nicht.
Und Stockenten nicht, wenn sie gerade gefressen haben.
Sonst schon!

Wohin mit "fliegen"?

(Nüchterne) Stockenten und Moorenten fliegen aber gleich! Code-Duplizierung!

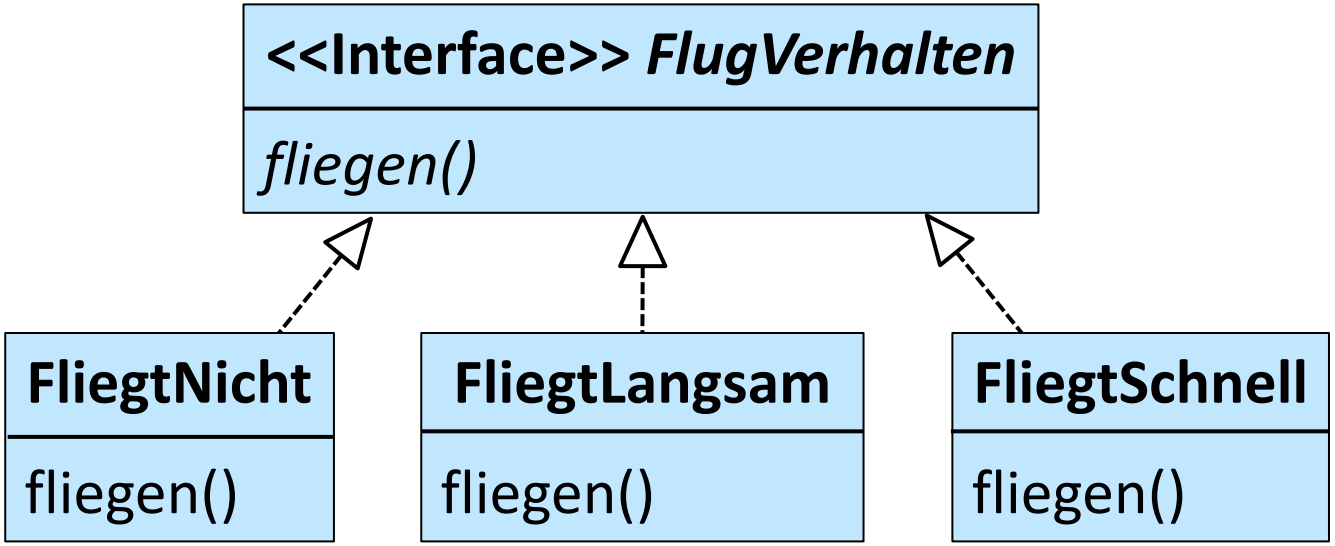


Lösung:

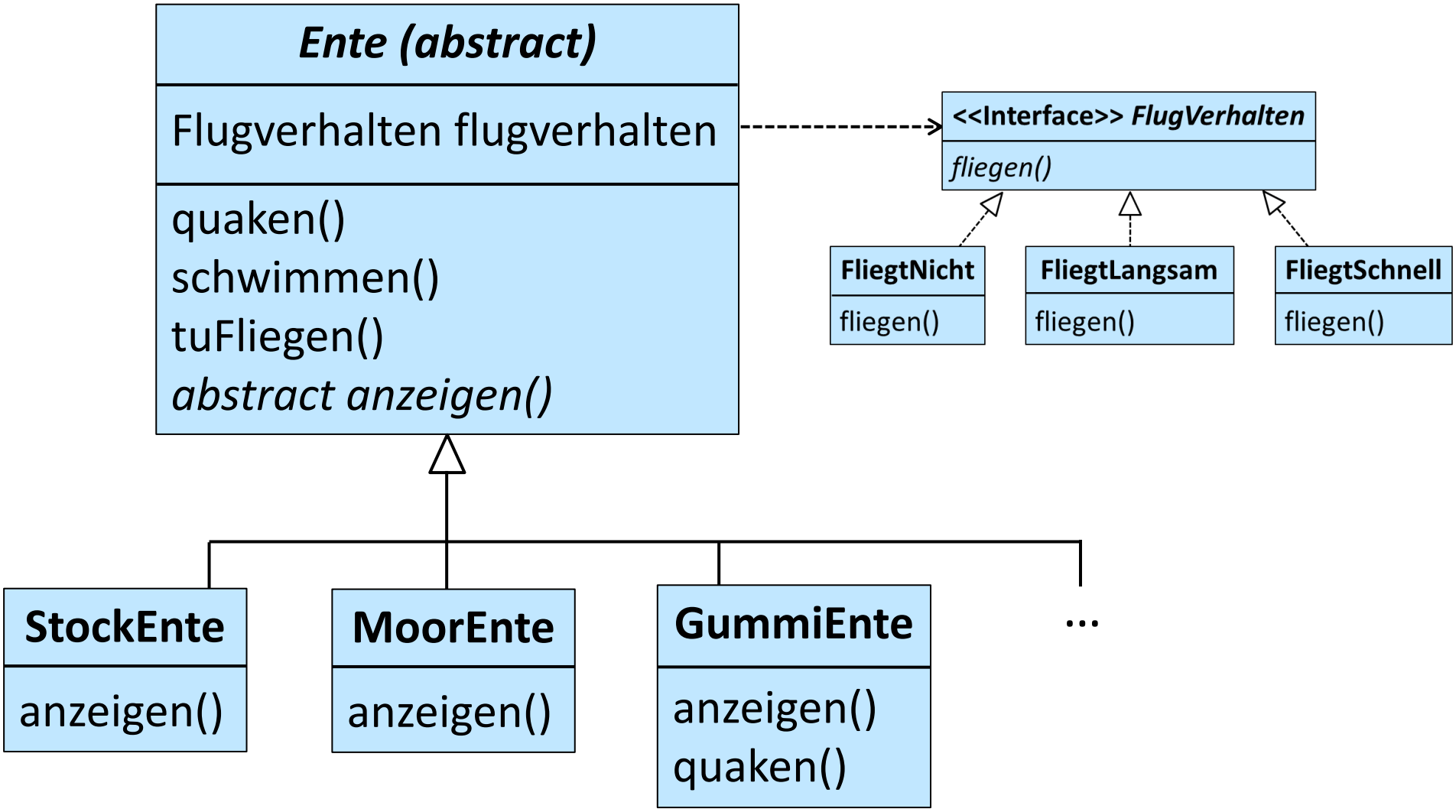
Das Veränderliche muss herausgezogen werden.

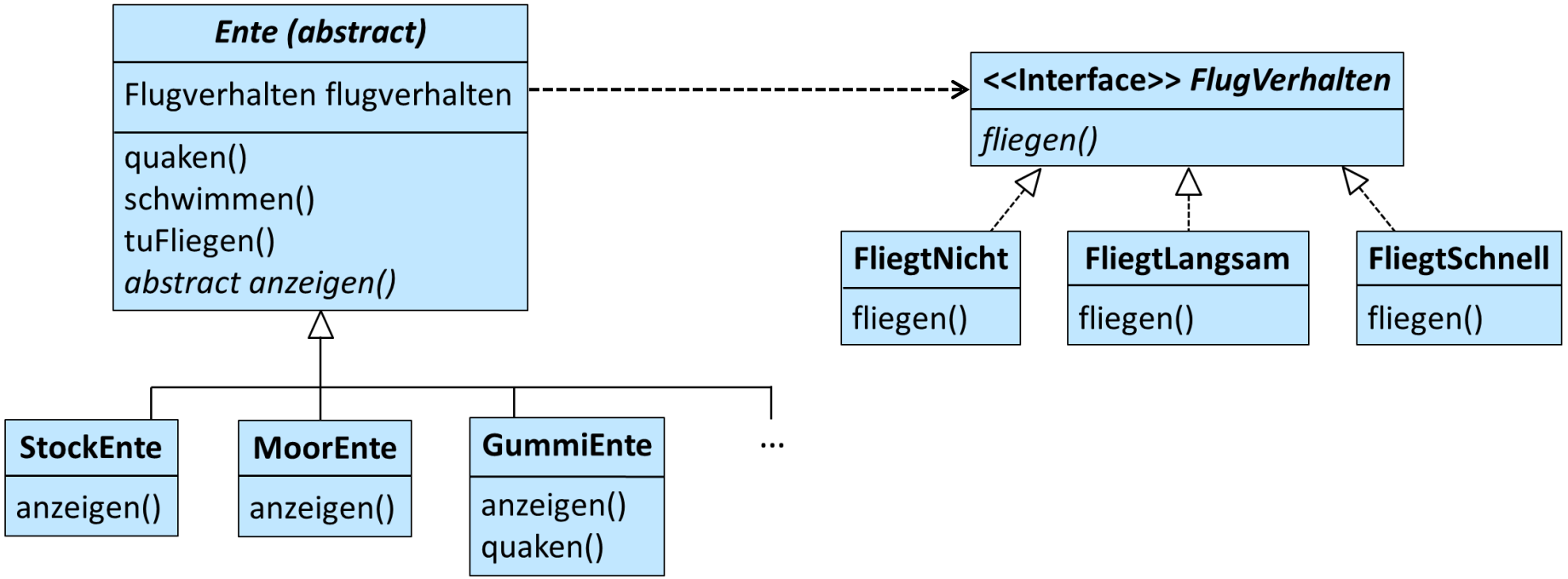
Hier: Das Flugverhalten

Gummienten können gar nicht fliegen, manche Entenarten können schnell fliegen, manche langsam, manche nicht, wenn sie satt sind.



Wie kommt die Ente nun zu ihrem Flugverhalten?

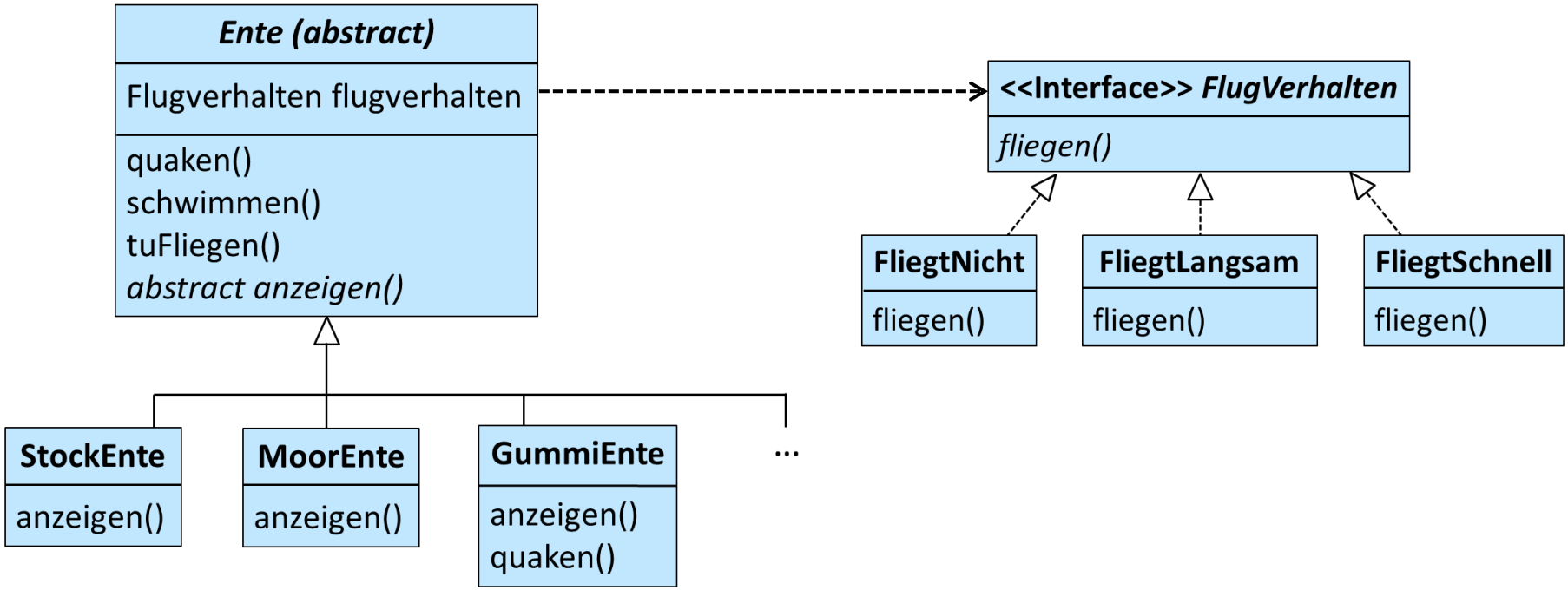

Ente (abstract)
Flugverhalten flugverhalten
quaken()
schwimmen()
tuFliegen()
abstract anzeigen()
<<Interface>> FlugVerhalten
fliegen()
FliegtNicht
fliegen()
FliegtLangsam
fliegen()
FliegtSchnell
fliegen()
StockEnte
anzeigen()
MoorEnte
anzeigen()
GummiEnte
anzeigen()
quaken()
...




```
public void tuFliegen(){
        flugverhalten.fliegen();
}
```

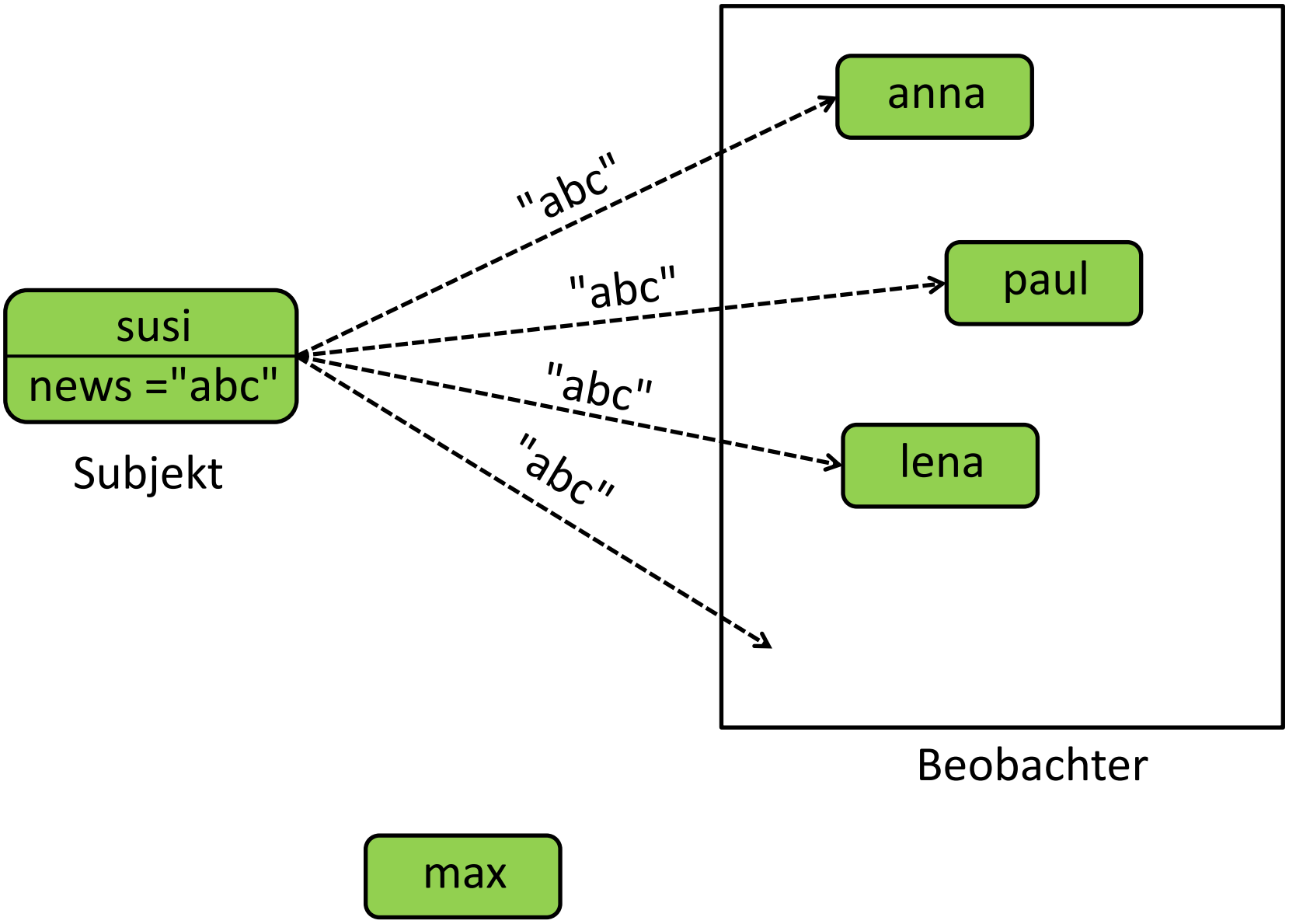
z.B. in der Klasse StockEnte:

```
private FlugVerhalten flugVerhalten = new FliegtSchnell();
```

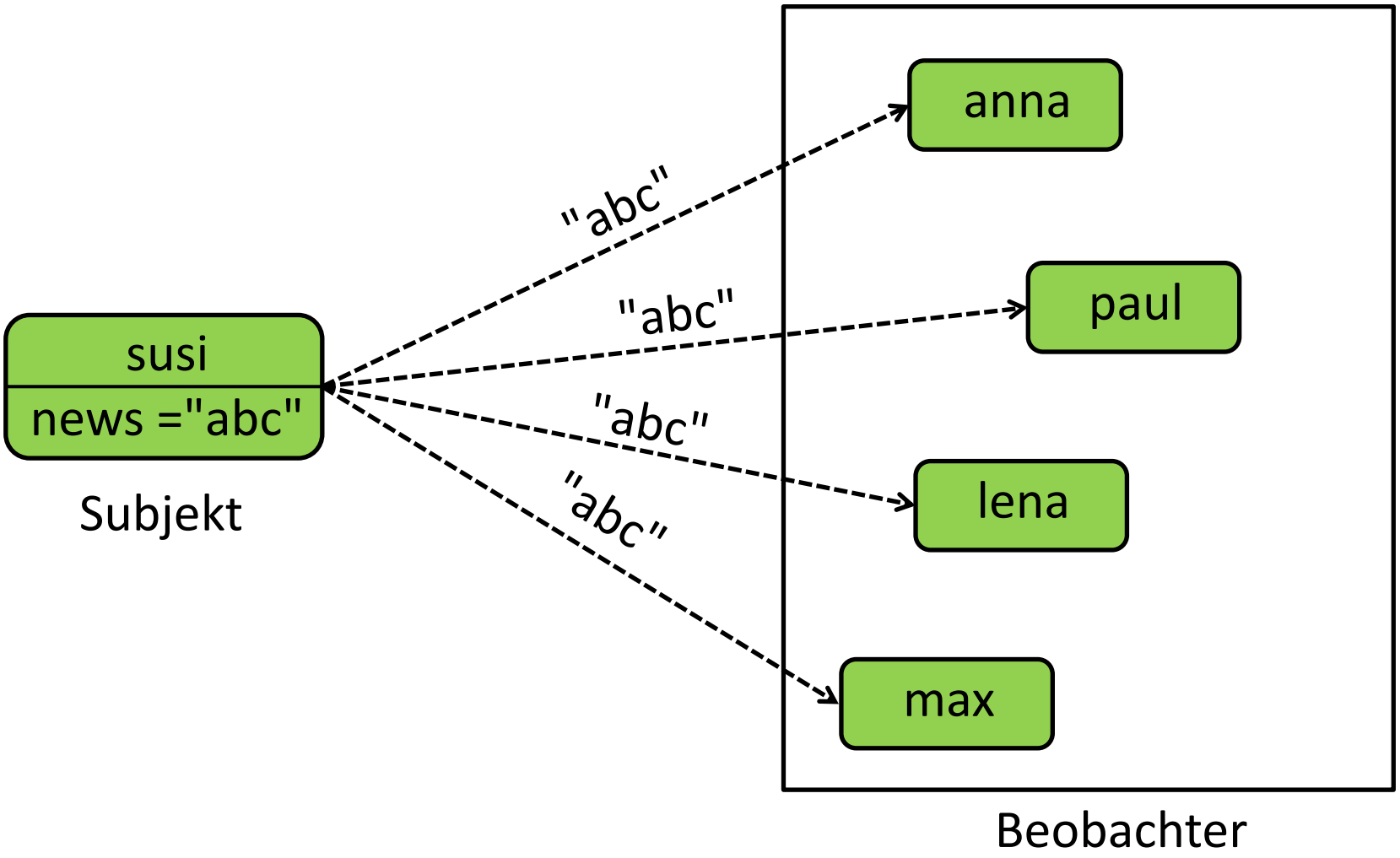



```
Ente donald = new StockEnte();
donald.tuFliegen();          ← fliegt schnell,
//donald frisst                wie alle Stockenten zunächst
donald.setFlugVerhalten(new FliegtNicht());
donald.tuFliegen();          ← fliegt nicht
//nach 5 Minuten
donald.setFlugVerhalten(new FliegtLangsam());
donald.tuFliegen();          ← fliegt langsam
```

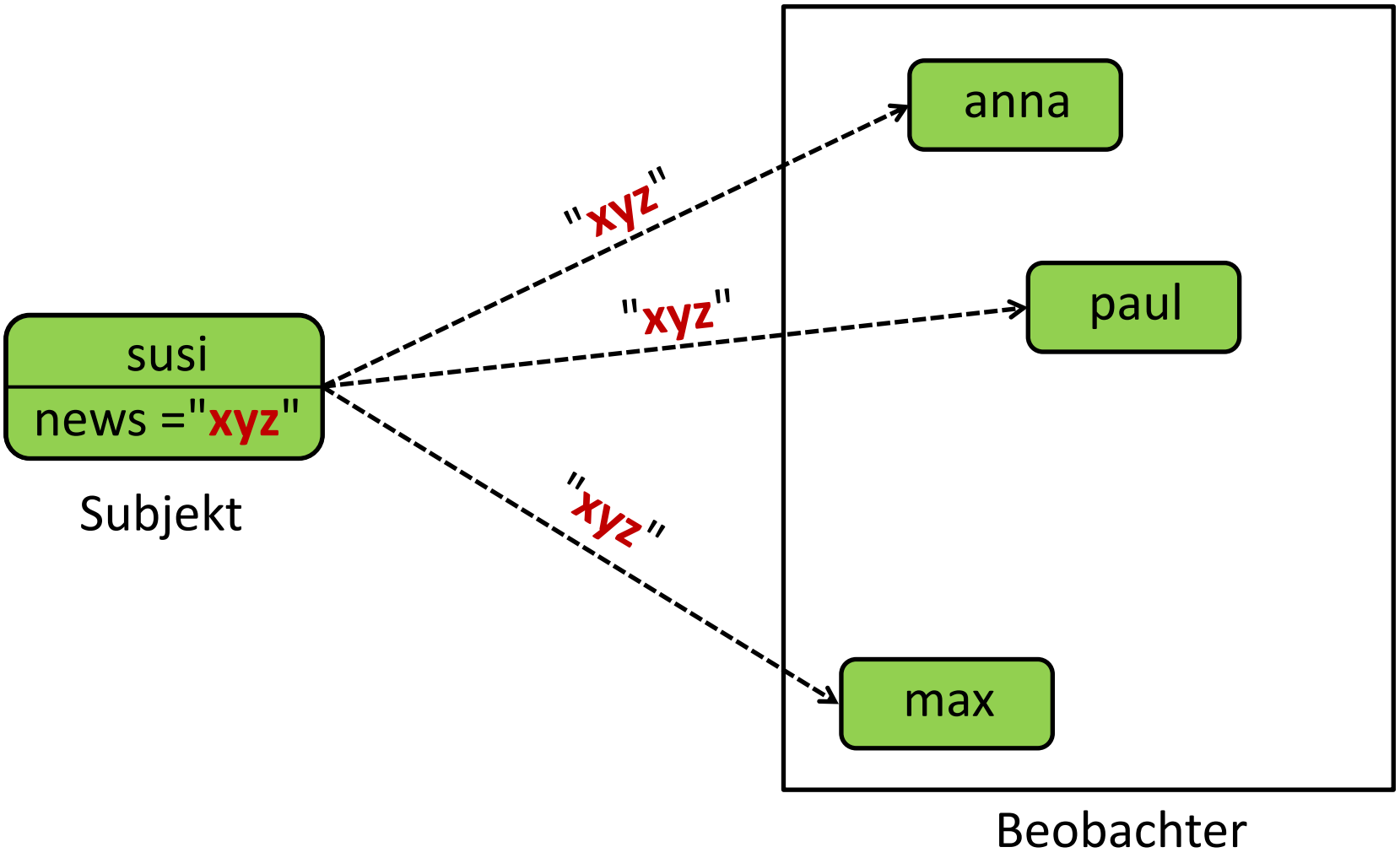
Beobachtermuster (observer pattern)



Beobachtermuster (observer pattern)

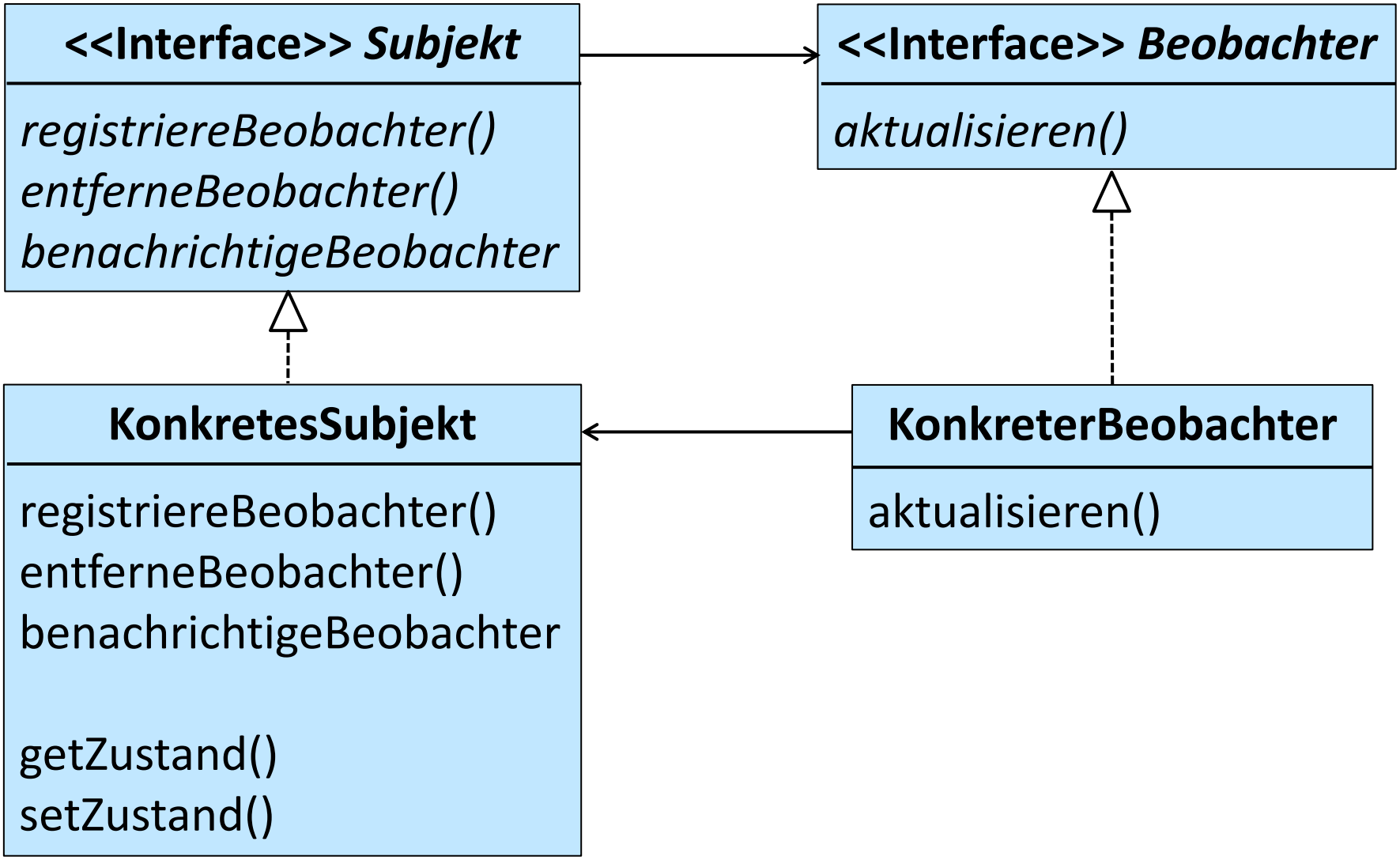


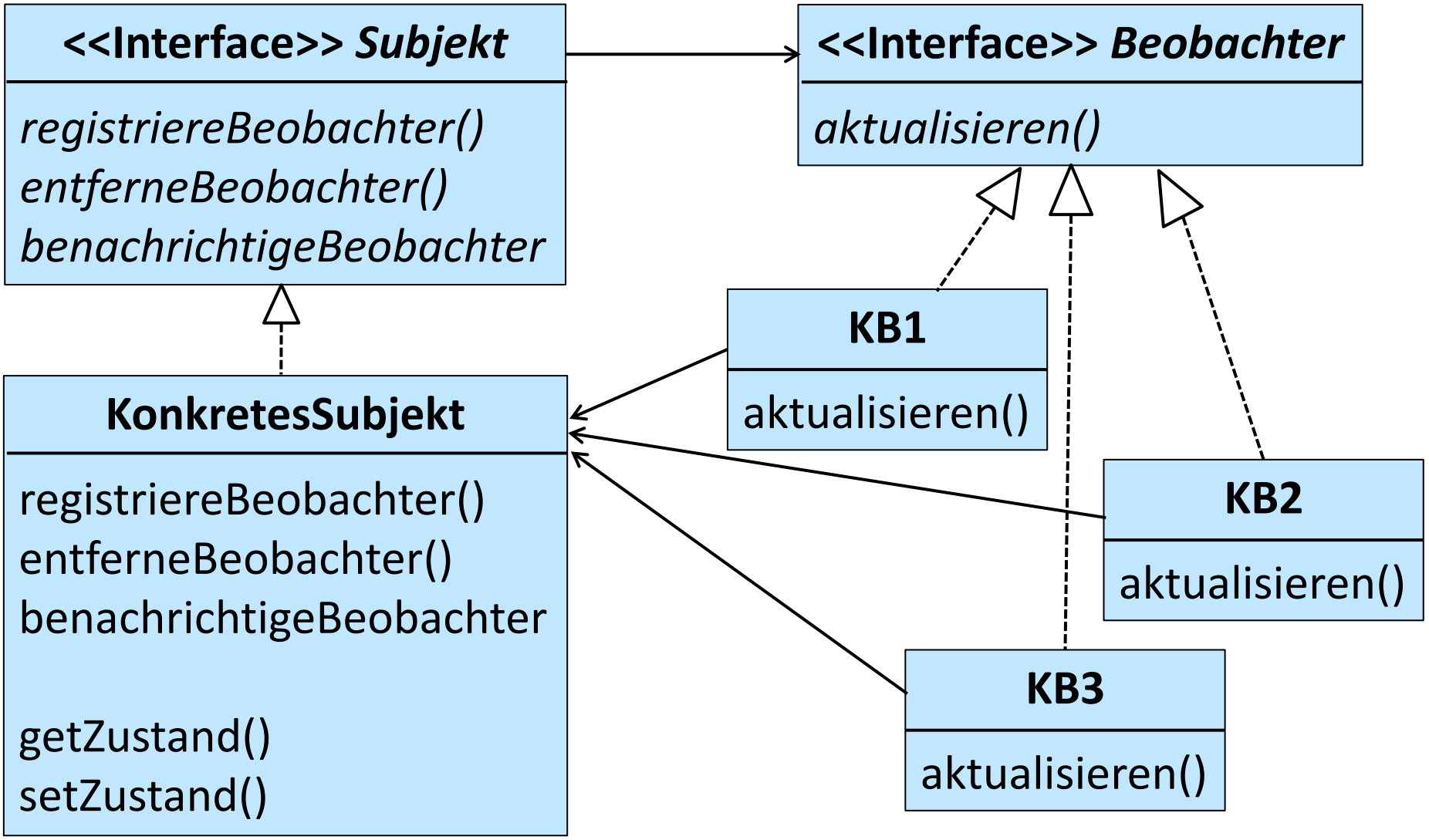
Beobachtermuster (observer pattern)



Beobachtermuster (observer pattern)

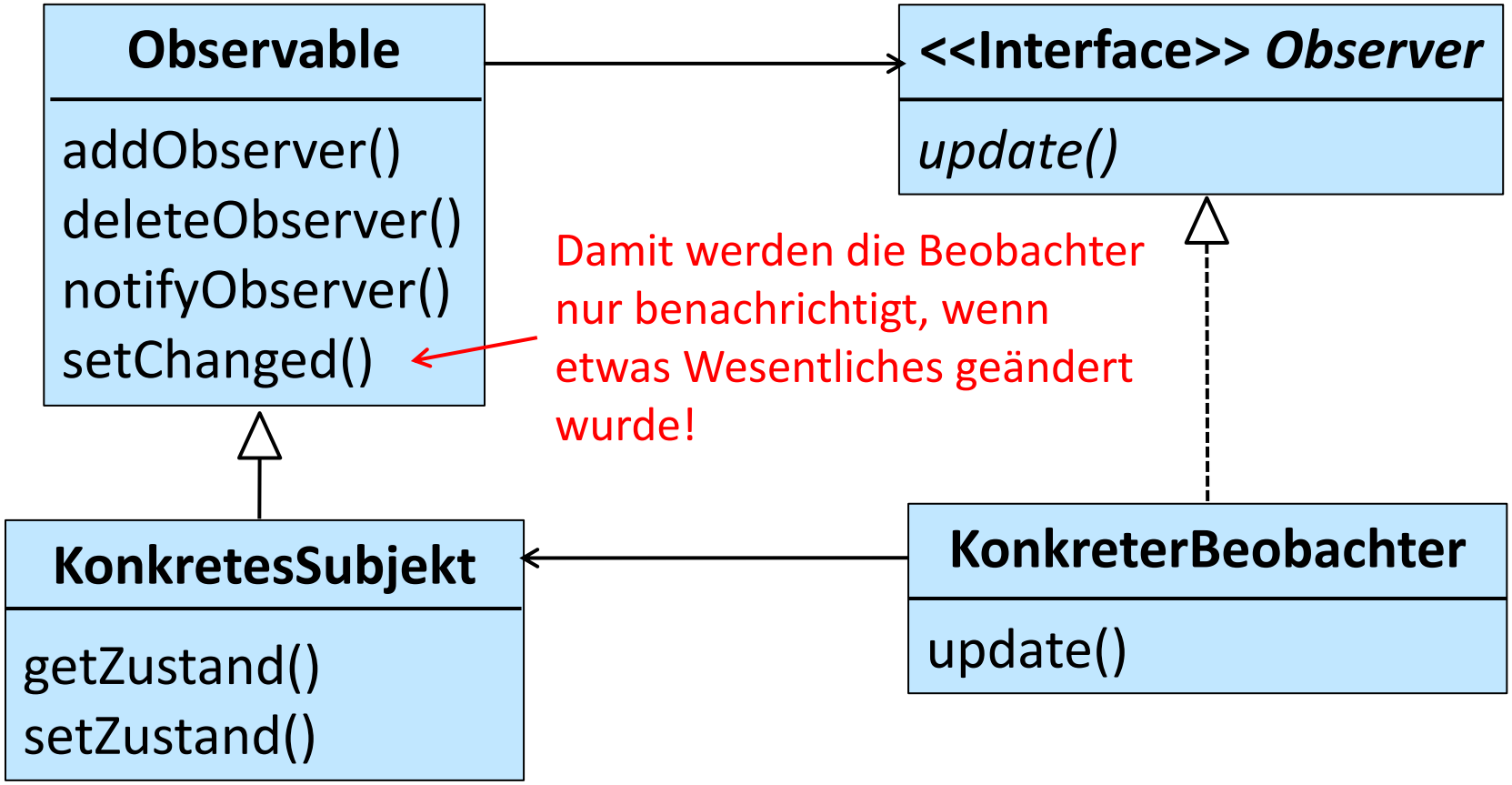
- strukturiert die (einseitige) Kommunikation zwischen einem Erzeuger von Information und den Interessenten an dieser Information (Beobachter).
- ermöglicht, dass sich Objekte (Observer, beobachtendes Objekt) bei einem anderem Objekt (Subject, beobachtetes Objekt) registrieren und fortan vom diesem informiert werden, sobald es sich ändert.
- Eins-zu-Viele-Beziehung zwischen Subjekt(Observable) und Beobachter-Objekten


<<Interface>> Subjekt
registriereBeobachter()
entferneBeobachter()
benachrichtigeBeobachter
<<Interface>> Beobachter
aktualisieren()
KonkretesSubjekt
registriereBeobachter()
entferneBeobachter()
benachrichtigeBeobachter
getZustand()
setZustand()
KonkreterBeobachter
aktualisieren()

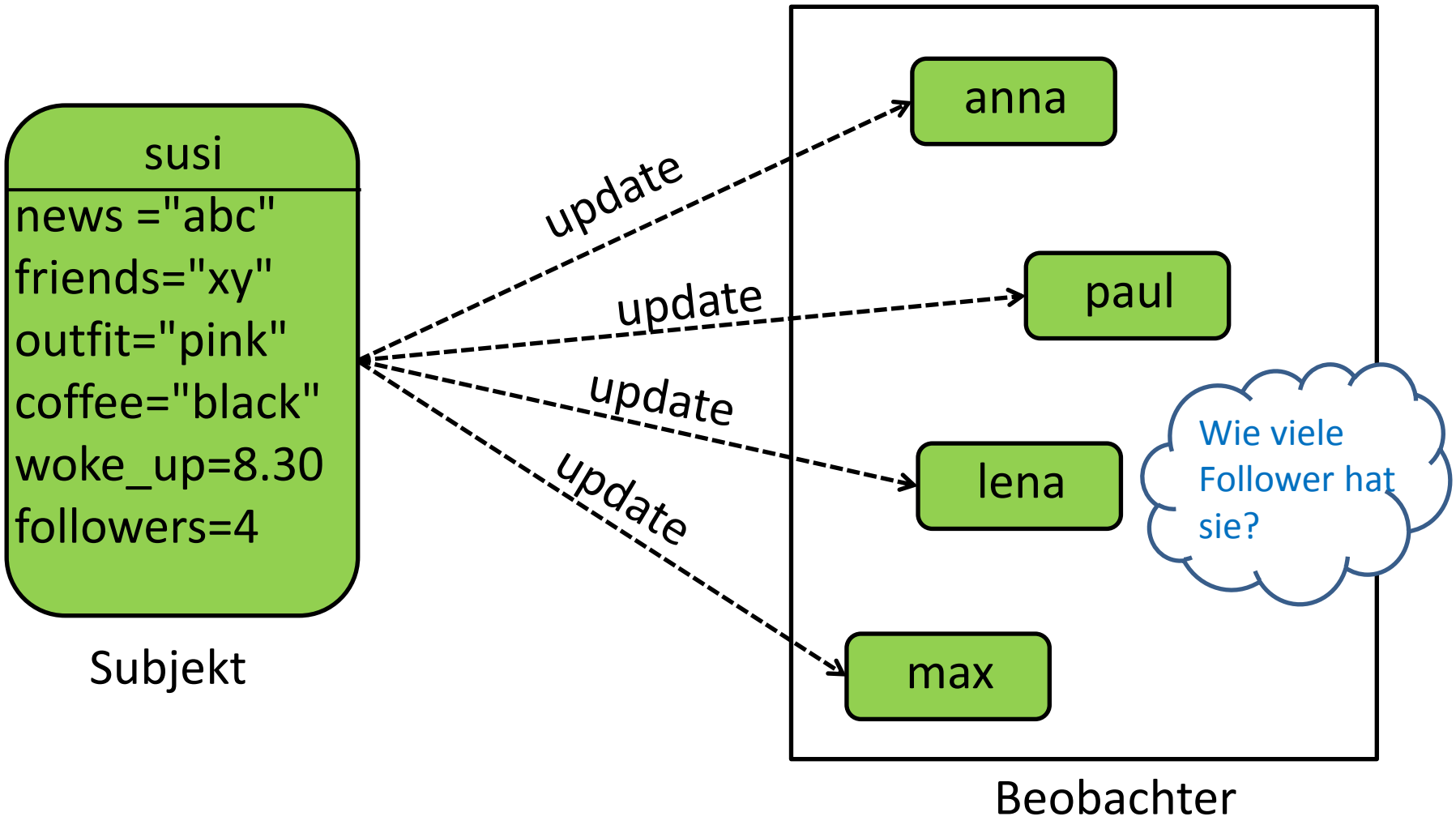


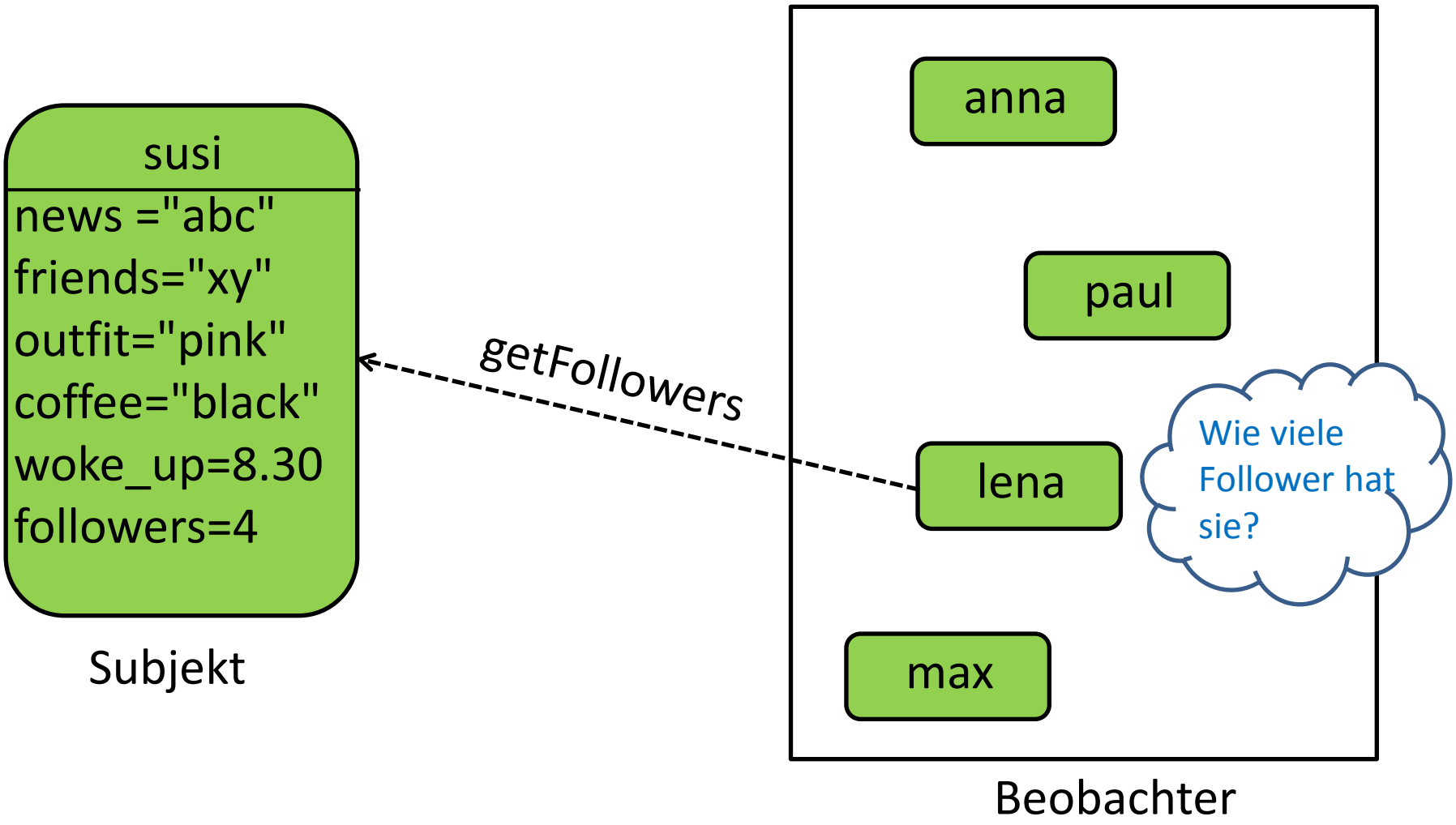
Das Subjekt muss die Klassen der konkreten Beobachter nicht näher kennen. Es reicht, wenn sie das Interface Beobachter implementieren und damit die Methode aktualisieren

Man kann statt des Interfaces Subjekt auch die schon vorhandene Klassen Observable und das Interface Observer benutzen. Dann braucht man sich um die Verwaltungsmethoden der Beobachterliste nicht mehr kümmern.



Nachteil:
Damit ist die einzige Vererbungsmöglichkeit ausgeschöpft!


susi
news ="abc"
friends="xy"
outfit="pink"
coffee="black"
woke_up=8.30
followers=4
Subjekt
update
update
update
update
anna
paul
lena
max
Wie viele Follower hat sie?
Beobachter



Mit update kann entweder ein ganzes Datenobjekt mitgegeben werden oder nichts. In dem Fall holen sich die Beobachter die gewünschten Daten selber.