Modellierung:
Prinzipien, CRC, DBC, Behaviour Protocols

Tutorium 3

6. März 2009

Schnittstellenspezifikation

- Identifikation: **CRC-Karten**
- Festlegung: Signatur-Spezifikation
- Verfeinerung: **Design by Contract**
- Verfeinerung: Interaktions-Spezifikation durch **Behaviour Protocols**

} Verhaltens-Spezifikation

Prinzipien

Keine Redundanzen im Modell!

Redundantes Modell

- Mehrfache Wege um die gleiche Information zu erhalten
- Speicherung abgeleiteter Informationen

Problem

- Bei Änderungen zur Laufzeit, müssen die Abgeleiteten Informationen / Objekte konsistent gehalten werden
- Zusatzaufwand bei Implementierung (Benachrichtigungsmechanismus) und zur Laufzeit (Benachrichtigung über Änderung und Aktualisierung abgeleiteter Infos)
- Änderungen im Design müssen evtl. an vielen Stellen nachgezogen werden

Behandlung

- Redundanz entfernen
- ... falls sie nicht als Laufzeitoptimierung unverzichtbar ist, da die Berechnung der abgeleiteten Informationen zu lange dauert

Prinzipien

Kein Bezug auf nicht-inhärente Klassen!

Problem:

Bezug auf **nicht**-inhärente Klassen führt unnötige Abhängigkeiten ein

Prinzipien

Nicht einheitliche Eigenschaften vermeiden!

Symptom

- bestimmte Eigenschaften (Methoden oder Variablen) einer Klasse sind nur für manche Instanzen gültig

Konsequenzen

- Abhängigkeit von bestimmten Fallunterscheidungen
- Unklare Funktionalität
- Wartung erschwert

Behandlung

- Klasse aufsplitten
- Evtl. Klasse einführen die „alle anderen Fälle“ darstellt

Prinzipien

Ersetzbarkeit

Problem:

Unterklasse hat eine stärkere Invariante als die Oberklasse

Fehlende Ersetzbarkeit

- In einem Kontext wo man Rotierbarkeit erwartet darf kein nicht-rotierbares Objekt übergeben werden

Prinzipien

Ersetzbarkeit und Kontrakte

B ist ein Subtyp von **A**

:⇔

Instanzen von **B** sind immer für Instanzen von **A** einsetzbar

⇔

Instanzen von **B** fordern höchstens und bieten mindestens
das gleiche wie Instanzen von **A**

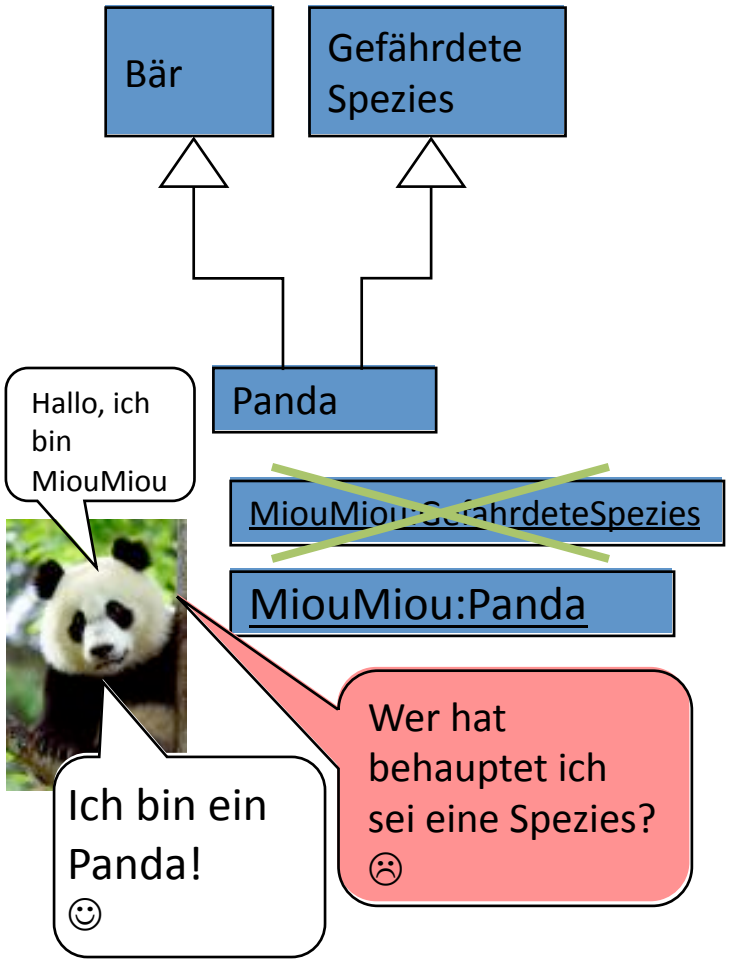
⇔

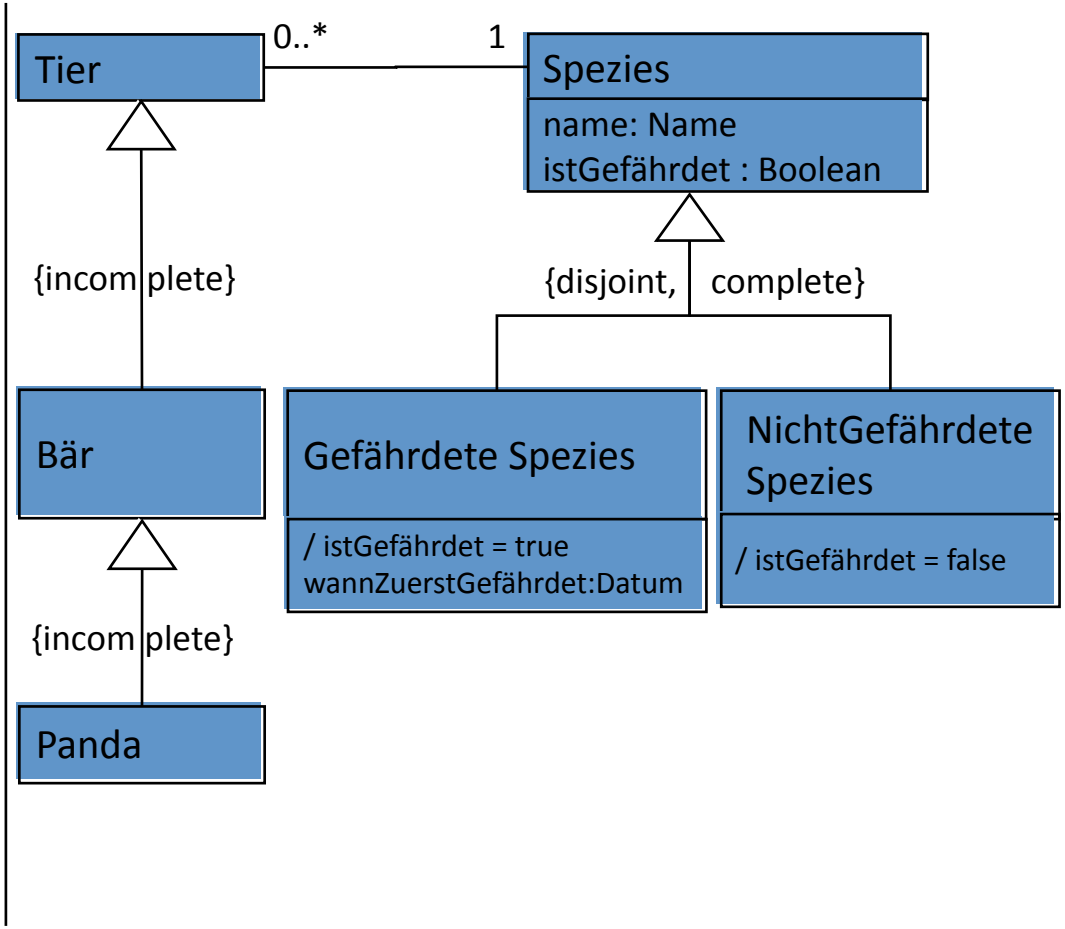
Instanzen von B haben mindestens alle Methoden von A,
und zwar mit (gleichen oder) stärkeren Nachbedingungen
und schwächeren Vorbedingungen

Prinzipien

Keine Verwirrung von Klassen und Instanzen!

Schlüssel zur Verständnis des Problems ist die Frage:
was sind die *Instanzen* dieser drei Klassen?





Prinzipien

Keine Verwirrung von Klassen und Instanzen!

Alternatives Kriterium

- Ersetzbarkeit
  - Bsp.:

    Kann ich "Miou-Miou" überall da einsetzen, wo ich eine Spezies erwarte?

Prinzipien

Abhängigkeiten vermeiden!

- Abhängigkeiten zwischen Kapselungseinheiten reduzieren
- Abhängigkeiten innerhalb der Kapselungseinheiten maximieren

Nutzen:

- Wartungsfreundlichkeit

Reduktion von Abhängigkeiten durch

„**Verbergen von Informationen**“

(„information hiding”)

Information Hiding

Reduktion von Abhängigkeiten

- **“Need to know” Prinzip** → „Schlanke Schnittstelle“

  Zielkonflikt:

  Verbergen von Informationen vs. Effizienz

- Verbergen von internen Objekten an den Grenzen des Subsystems
- Verberge Datenstrukturen einer Klasse
- Führe eine Operation nicht auf dem Ergebnis einer anderen aus. → **„Law of Demeter“ Prinzip**

  Zielkonflikt:

  Verbergen von Informationen vs. „schlanke“ Schnittstellen.

Prinzipien

Law of Demeter

„Talk only to your friends!“

- Klasse sollte nur von „Freunden“, d.h. den Typen der eigenen Felder, Methoden- und Ergebnisparameter abhängig sein.
- Insbesondere sollte sie nicht Zugriffsketten nutzen, die Abhängigkeiten von den Ergebnistypen von Methoden der Freunde erzeugen.

Beispiel:

- Nicht: param.m().mx().my()….;
- Sondern: param.mxy();

  wobei die methode mxy() den transtiven Zugriff kapselt.

Prinzipien

(Design Principles, **Robert C. Martin**, 1996)

- Dependency Inversion Principle (DIP)
- Acyclic Dependencies Principle (ADP)
- Stable Dependencies Principle (SDP)
- Stable Abstractions Principle (SAP)

Quellenangabe

- Dr. Kniesel, *Vorlesungsfolien zu Softwaretechnologie*, WS 2008.
- Meilir Page-Jones, *Fundamentals of Object-Oriented Design in UML*, Addison-Wesley, 2000.