Programmierkonzepte in der Physikalischen Chemie

25. Liskov-Substitutionsprinzip

UNI FREIBURG

Albert-Ludwigs-Universität Freiburg

Dr. Till Biskup

Institut für Physikalische Chemie
Albert-Ludwigs-Universität Freiburg
Wintersemester 2016/17

Zentrale Aspekte

UNI FREIBURG

- Objekte einer abgeleiteten Klasse sollten sich genauso verhalten wie Objekte der Basisklasse.
- Verletzungen des Liskov-Substitutionsprinzips lassen sich immer nur im konkreten Kontext feststellen.
- Erwartungen an das Verhalten lassen sich in den wenigsten Sprachen explizit im Code formulieren.
- Das Liskov-Substitutionsprinzip ist eine wesentliche Voraussetzung für das Open-Closed-Prinzip.
- „Entwurf gemäß Vereinbarung“ (*design by contract*) formalisiert die Erwartungen an eine Klasse.

Übersicht

UNI FREIBURG

Das Liskov-Substitutionsprinzip

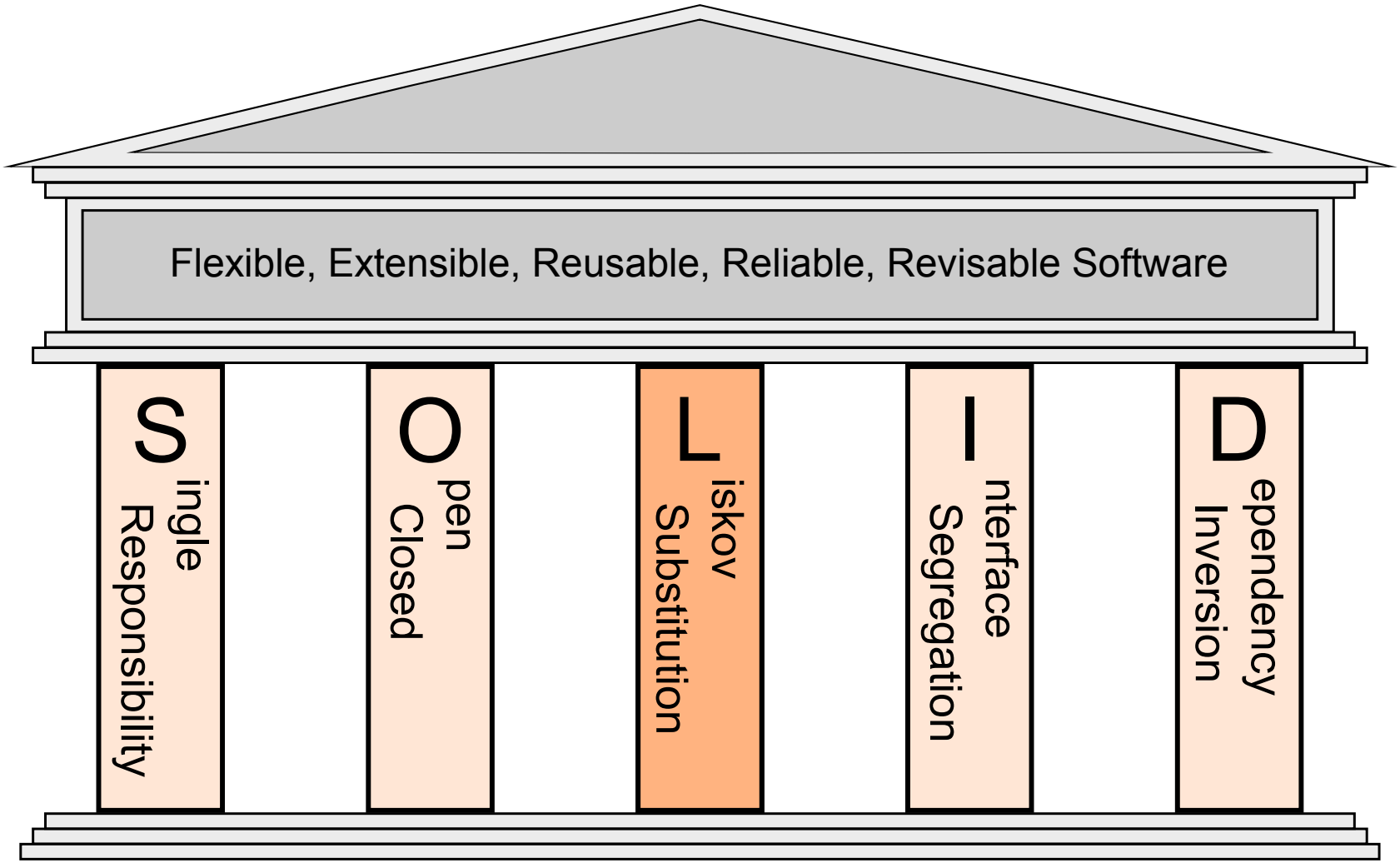
Symptome, die für seinen Einsatz sprechen

Beispiele für seinen Einsatz

Bedeutung im Gesamtkontext der Software-Architektur

Das Liskov-Substitutionsprinzip

Übersicht über die fünf Prinzipien



Das Liskov-Substitutionsprinzip

Eine verhaltensbasierte Definition von Subtypen

UNI FREIBURG

> *objects of the subtype ought to behave the same as those of the supertype as far as anyone or any program using supertype objects can tell.*
>
> – Liskov und Wing

- Typen sind im Kontext der OOP Klassen.
- Das erwartete Verhalten von Objekten steht im Fokus.
  - Ein Subtyp sollte sich genauso wie ein Grundtyp verhalten.
  - Ein Subtyp kann zusätzliches Verhalten aufweisen.

☛ Hinweise und Regeln für Vererbungshierarchien, die das Open-Closed-Prinzip ermöglichen

Liskov und Wing, *ACM Trans. Program. Lang. Syst.* **16**:1811–1841, 1994

Das Liskov-Substitutionsprinzip

Regel für Vererbungshierarchien

UNI FREIBURG

- Hängt eng mit dem Open-Closed-Prinzip zusammen.
  - „Eine Klasse sollte offen für Erweiterung, aber verschlossen gegenüber Änderungen sein.“
  - Abgeleitete Klassen implementieren abstrakte Methoden, die in Basisklassen definiert wurden.
  - erfordert klare Regeln für die Vererbung
- Vererbung wird oft als *ist ein*-Beziehung gesehen.
  - Das kann konzeptionell zu Problemen führen.
- Fragen, die adressiert werden:
  - Was muss man beim Einsatz von Vererbung beachten?
  - Welche Probleme können bei der Vererbung auftreten, die das Open-Closed-Prinzip verletzen?

Symptome, die für seinen Einsatz sprechen

Zerbrechlichkeit und Verletzung des Open-Closed-Prinzips

UNI FREIBURG

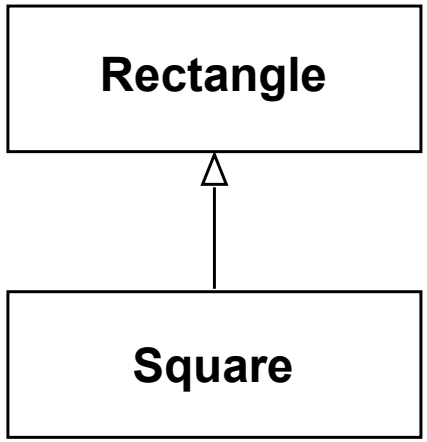
- Zerbrechlichkeit
  - Einsatz eines Objekts einer Subklasse statt eines Objekts der Superklasse führt zu unerwartetem Verhalten.
- Verletzung des Open-Closed-Prinzips
  - „Behebung“ des Problems der Nichtersetzbarkeit (s.o.): Abfrage des spezifischen Typs eines Objekts
  - Explizite Überprüfung des (Sub-)Typs erzwingt Codeänderungen mit jedem neuen Subtyp.

☛ Verletzungen des Liskov-Substitutionsprinzips lassen sich immer nur im konkreten Kontext feststellen.

☛ Erwartungen an das Verhalten lassen sich in den wenigsten Sprachen explizit im Code formulieren.

Beispiele für seinen Einsatz

Rechtecke und Quadrate



- `Rectangle` ist die Superklasse, `Square` die Subklasse.
  - Vererbung führt zu einer *ist-ein*-Beziehung.
  - *Geometrisch* ist ein Quadrat ein Rechteck.
- Entspricht die Relation dem Liskov-Substitutionsprinzip?
  - hängt vom Kontext und vom erwarteten Verhalten ab

Beispiele für seinen Einsatz

Rechtecke und Quadrate

UNI FREIBURG

Situation 1: Rechteck und Quadrat sind unveränderlich

- Höhe und Breite können nicht verändert werden.
  - nur Getter, aber keine Setter vorhanden
  - Höhe und Breite werden im Konstruktor gesetzt.
- Weiteres Verhalten ist nicht betroffen.
  - Bsp.: Fläche kann berechnet werden.
  - Darstellung etc. ist ebenfalls invariant.

☛ `Square` kann statt `Rectangle` verwendet werden, ohne dass anderer Code geändert werden müsste.

☛ Keine Verletzung des Liskov-Substitutionsprinzips.

Situation 2: Rechteck und Quadrat sind veränderlich

- Höhe und Breite können verändert werden.
  - Höhe und Breite sind im Rechteck unabhängige Größen.
  - Höhe und Breite sind im Quadrat immer identisch.
- Das Verhalten ist *unterschiedlich*.
  - Rechteck: Seiten sollten *unabhängig* änderbar sein.
  - Quadrat: Seiten sind immer gekoppelt – Änderung der Breite führt zwangsläufig zur Änderung der Höhe.

☛ `Square` kann *nicht* statt `Rectangle` verwendet werden, ohne dass es zu Überraschungen kommen kann.

☛ *Verletzung* des Liskov-Substitutionsprinzips.

Heuristiken und Konventionen

Hinweise auf Verletzungen des Liskov-Substitutionsprinzips

UNI FREIBURG

Grundregel

Eine abgeleitete Klasse, die weniger tut als ihre Basisklasse, kann meist nicht anstelle der Basisklasse eingesetzt werden.

- degenerierte Funktionen in abgeleiteten Klassen
  - Grund: Funktion in der abgeleiteten Klasse nutzlos
  - führt nicht immer, aber oft zu Problemen
  - Gleichzeitig Symptom für Interface-Segregation-Prinzip
- neue Ausnahmen (*exceptions*) in abgeleiteten Klassen
  - Vorbedingungen dürfen für abgeleitete Klassen nicht strikter sein als für die Basisklasse.

Bedeutung im Gesamtkontext

Voraussetzung für das Open-Closed-Prinzip

UNI FREIBURG

- legt Kriterien für Vererbungsbeziehungen fest
  - Austauschbarkeit abgeleiteter Klassen mit der Basisklasse
  - gleiches Verhalten aus Sicht der Basisklasse
- ermöglicht das Open-Closed-Prinzip
  - beruht auf Vererbung von abstrakten Klassen
  - Die abstrakte Basisklasse definiert das Verhalten, an das sich die abgeleiteten Klassen halten müssen.
  - führt zu Flexibilität, Wiederverwendbarkeit, Wartbarkeit
- „Entwurf kraft Vereinbarung“ (*design by contract*)
  - formalisiert die Erwartungen an eine Klasse und die von ihr abgeleiteten Klassen
  - lässt sich in den wenigsten Sprachen direkt formulieren

Formalisierung der Erwartungen an eine Klasse

„Entwurf kraft Vereinbarung“ (*design by contract*)

UNI FREIBURG

- zwei Arten von Erwartungen
  - Vorbedingung: Voraussetzung für einen Methodenaufruf
  - Nachbedingung: garantierter Zustand nach dem Aufruf
- Anforderungen an abgeleitete Klassen
  - dürfen keine strengeren Vorbedingungen fordern
  - müssen mindestens die von der Basisklasse vorgegebenen Nachbedingungen erfüllen
- Formulierung von Erwartungen
  - in den wenigsten Sprachen direkt möglich
  - Unittests sind eine gute Möglichkeit
  - ggf. in der Basisklasse explizit dokumentieren

Zentrale Aspekte

UNI FREIBURG

- Objekte einer abgeleiteten Klasse sollten sich genauso verhalten wie Objekte der Basisklasse.
- Verletzungen des Liskov-Substitutionsprinzips lassen sich immer nur im konkreten Kontext feststellen.
- Erwartungen an das Verhalten lassen sich in den wenigsten Sprachen explizit im Code formulieren.
- Das Liskov-Substitutionsprinzip ist eine wesentliche Voraussetzung für das Open-Closed-Prinzip.
- „Entwurf gemäß Vereinbarung“ (*design by contract*) formalisiert die Erwartungen an eine Klasse.