

Programmierkonzepte in der Physikalischen Chemie

26. Interface-Segregation-Prinzip

Albert-Ludwigs-Universität Freiburg



**UNI
FREIBURG**

Dr. Till Biskup

Institut für Physikalische Chemie
Albert-Ludwigs-Universität Freiburg
Wintersemester 2017/18



- ❏ Keine Klasse sollte Methoden implementieren müssen, die sie nicht nutzt.
- ❏ Schnittstelle und Nutzer können beide Veränderungen der jeweils anderen Seite erzwingen.
- ❏ Klassen können mehrere Schnittstellen für unterschiedliche Nutzer erfordern.
- ❏ Benötigt eine Klasse mehrere Schnittstellen, werden sie als separate, abstrakte Klassen definiert.
- ❏ Die Entkopplung macht den Code übersichtlicher und sorgt für Flexibilität, Wiederverwendbarkeit, Wartbarkeit.

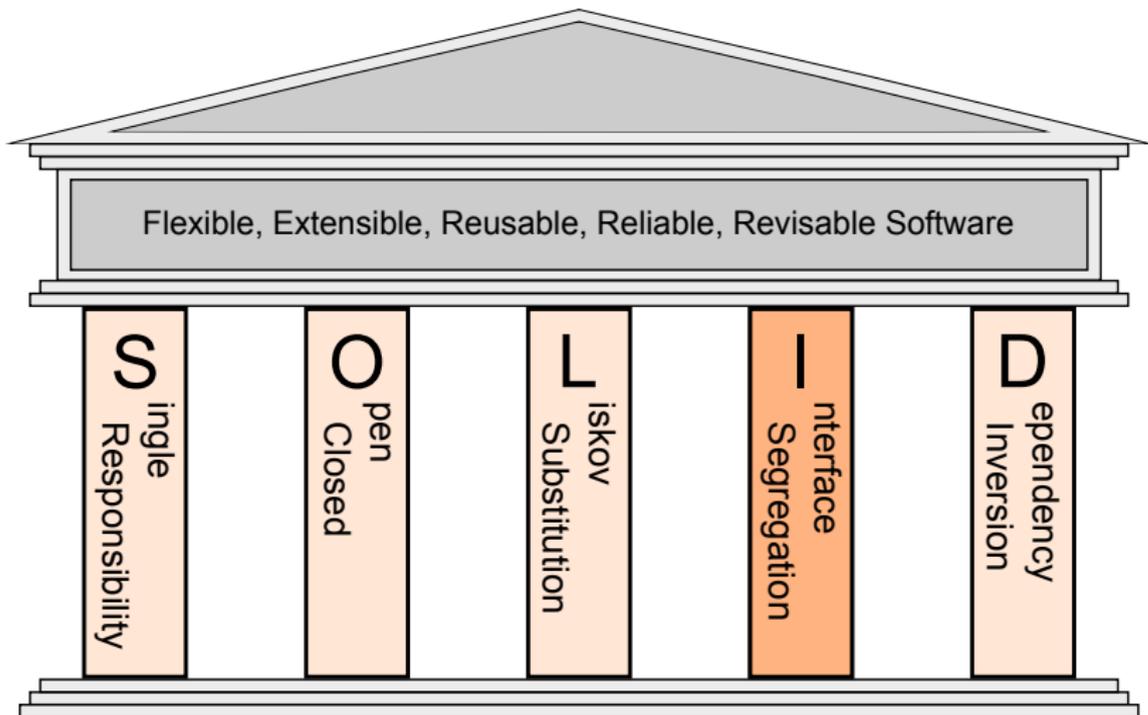
Das Interface-Segregation-Prinzip

Beispiele für seinen Einsatz

Bedeutung im Gesamtkontext der Software-Architektur

Das Interface-Segregation-Prinzip

Übersicht über die fünf Prinzipien



“ *Clients should not be forced to depend on methods that they do not use.*

– Robert C. Martin

- ▶ Schnittstellen abstrakter Klassen sollten minimal sein.
 - Die Methoden der abstrakten Klasse sollten *alle* in *jeder* abgeleiteten Klasse verwendet werden.
- ▶ Kontext: Kohäsion
 - Elemente eines Moduls gehören funktional zusammen.
- ▶ Trennung von Verantwortlichkeiten
 - Änderungen sollten keine Auswirkungen auf konzeptionell getrennte Teile des Programms haben.

- ▶ Schnittstellen und Nutzer beeinflussen sich gegenseitig.
 - Normale Sicht auf Abhängigkeiten: Änderungen der Schnittstelle zwingen deren Nutzer zu Änderungen.
 - Aber: Nutzer können ihrerseits Änderungen einer Schnittstelle erzwingen.
- ▶ Objekte können mehr als eine Schnittstelle benötigen.
 - Jede Schnittstelle sollte als einzelne abstrakte Klasse mit zusammengehörenden Methoden definiert werden.
 - Nutzung über Mehrfachvererbung bzw. Implementierung von Schnittstellen (*interfaces*)
- ▶ anderer Blickwinkel als das SRP
 - Das ISP fokussiert auf die Auswirkungen auf Unbeteiligte statt auf die Gründe für die Veränderung.

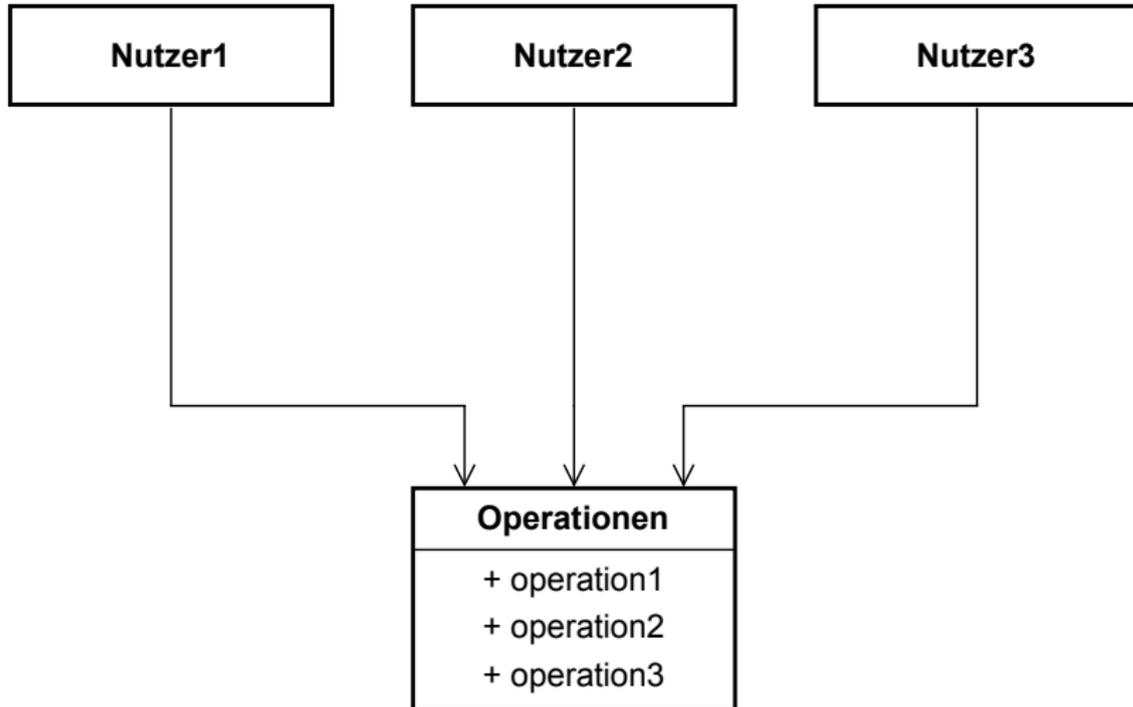
Kapselung (*encapsulation*)

Die Kommunikation mit einem Objekt erfolgt ausschließlich über eine minimale öffentliche Schnittstelle, die keine Interna der Implementierung „verrät“.

- ▶ Grundlage der Modularität und Austauschbarkeit
 - essentiell für Flexibilität, Wiederverwendbarkeit, Wartbarkeit
- ▶ Spannungsfeld Vererbung–Kapselung
 - Vererbung bricht Kapselung
 - LSP und ISP: Kriterien für korrekten Einsatz der Vererbung
 - Fokus auf Modularität und Wiederverwendbarkeit

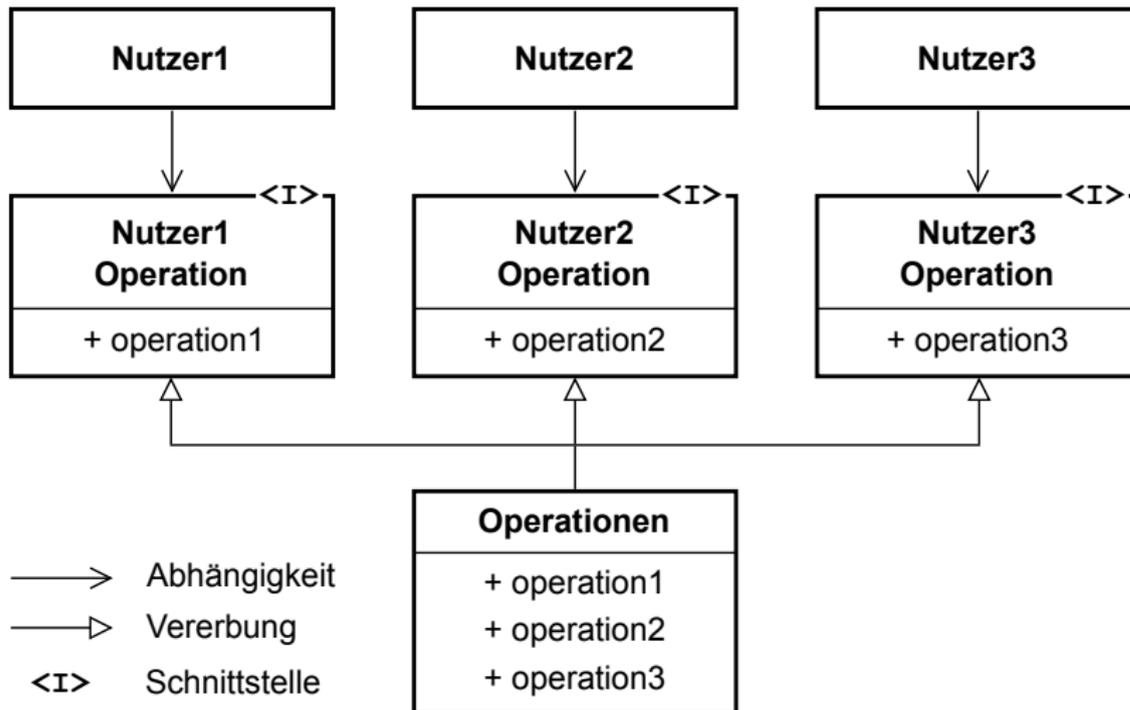
Beispiele für seinen Einsatz

Das abstrakte Beispiel: direkter Aufruf führt zur Kopplung



Beispiele für seinen Einsatz

Das abstrakte Beispiel: Entkopplung durch (abstrakte) Schnittstellen



- ▶ Entkopplung
 - Einzelne Module lassen sich wiederverwenden.
 - Änderungen der Funktionalität bleiben lokal.

- ▶ übersichtlicherer Code
 - Klassen implementieren nur das Notwendige.
 - Fokussierung und Kohäsion
 - erhöht die Lesbarkeit und damit die Wartbarkeit

- ▶ Zusammenhang mit dem SRP
 - Strategie zum Umgang mit Klassen, die mehr als eine Aufgabe erfüllen müssen
 - Jede Einzelaufgabe wird in eine Schnittstelle ausgelagert.

- ☞ sorgt für Flexibilität, Wiederverwendbarkeit, Wartbarkeit

“ *Don't force users of a component to depend on things they don't need.*

– Robert C. Martin

- ▶ Was gehört zusammen in eine Komponente?
 - Klassen und Module, die zusammen verwendet werden: Teile einer wiederverwendbaren Abstraktion
 - Kennzeichen: viele Abhängigkeiten untereinander
- ▶ Was gehört *nicht* zusammen in eine Komponente?
 - Klassen und Module, die sich unabhängig voneinander verwenden lassen (schwach gekoppelt sind)
- 👉 „Hänge nicht von Dingen ab, die du nicht brauchst.“



- ❏ Keine Klasse sollte Methoden implementieren müssen, die sie nicht nutzt.
- ❏ Schnittstelle und Nutzer können beide Veränderungen der jeweils anderen Seite erzwingen.
- ❏ Klassen können mehrere Schnittstellen für unterschiedliche Nutzer erfordern.
- ❏ Benötigt eine Klasse mehrere Schnittstellen, werden sie als separate, abstrakte Klassen definiert.
- ❏ Die Entkopplung macht den Code übersichtlicher und sorgt für Flexibilität, Wiederverwendbarkeit, Wartbarkeit.