Entwurfsmuster

Martin Fesser
OOIN

Inhalt

- Was sind Entwurfsmuster?
- Vorteile, Nachteile
- Entwurfsmusterkatalog (nach GoF)
- Variation von Entwurfsaspekten
- Wie Entwurfsmuster Entwurfsprobleme lösen
- Beispiele
- Fazit
- Quellen, weiterführende Literatur

Was sind Entwurfsmuster?

„Jedes Muster beschreibt ein in unserer Umwelt beständig wiederkehrendes Problem und erläutert den Kern der Lösung für dieses Problem, so daß Sie diese Lösung beliebig oft anwenden können, ohne sie jemals ein zweites Mal gleich auszuführen.“

Christopher Alexander: A Pattern Language, 1977

Was sind Entwurfsmuster? (Forts.)

- wiederkehrende Probleme immer auf die gleiche Art lösen
- Muster = Lösungsbeschreibung eines Problems
- Lösung möglichst flexibel und wiederverwendbar
- Zurückgreifen auf Expertenwissen, nicht versuchen Rad neu zu erfinden!
- Beschreibung eines Musters durch
  - Name
  - Problem
  - Lösung
  - Konsequenzen

Was sind Entwurfsmuster nicht?

- Algorithmen und Datenstrukturen (beziehen sich auf ein spezifisches Problem)
- Frameworks (sprachspezifische Klassensammlung)
- Allheilmittel zur Überwindung der Software-Krise

Vorteile

- bessere Wiederverwendbarkeit
- höheres Maß an Flexibilität
- Expertenwissen wird gesammelt
- einfachere Kommunikation zwischen Entwicklern
- bessere Dokumentation
- leichtere Wartung
- geringerer Zeitaufwand bis zum „richtigen“ Entwurf

Nachteile

- zum Teil unnötig hohe Kosten, zu viele Klassen
- hoher Lern- und Suchaufwand
- hoher Entwicklungsaufwand
- nicht automatisierbar

Entwurfsmusterkatalog

- nach E. Gamma, R. Helm, R. Johnson, J. Vlissides (Gang of Four, GoF)
- einheitliches Format:
  - Mustername
  - Zweck
  - auch bekannt als
  - Motivation
  - Anwendbarkeit
  - Struktur
  - Teilnehmer
  - Interaktionen
  - Konsequenzen
  - Implementierung
  - Beispielcode
  - bekannte Verwendungen
  - verwandte Muster

Unterteilung

- Klassifizierung in
  - Erzeugungsmuster
  - Strukturmuster
  - Verhaltensmuster
- Unterscheidung zwischen
  - klassenbasiert (statische Vererbung)
  - objektbasiert (dynamische Objektbeziehungen)

Erzeugungsmuster

- betreffen den Prozeß der Objekterzeugung
- klassenbasiert
  - Teile der Objekterzeugung verlagert in Unterklassen
- objektbasiert
  - Objekterzeugung wird an anderes Objekt delegiert

Strukturmuster

- befassen sich mit Zusammensetzung von Objekten und Klassen
- klassenbasiert
  - Vererbung, um Klassen zusammenzuführen
- objektbasiert
  - beschreibt Wege, Objekte zusammenzuführen

Verhaltensmuster

- charakterisieren Art und Weise, in der Klassen und Objekte zusammenarbeiten und Zuständigkeiten aufteilen
- klassenbasiert
  - Vererbung, um Algorithmen und Kontrollfluß zu beschreiben
- objektbasiert
  - beschreibt, wie Gruppe von Objekten zusammenarbeitet um Aufgabe zu erfüllen, die einzelnes Objekt allein nicht erfüllen kann

Übersicht

		Aufgabe		
		Erzeugungsmuster	**Strukturmuster**	**Verhaltensmuster**
Gültigkeits-bereich	**klassen-basiert**	Fabrikmethode	Adapter (klassenbasiert)	Interpreter Schablonenmethode
	objekt-basiert	Abstrakte Fabrik Erbauer Prototyp Singleton	Adapter (objektbasiert) Brücke Dekorierer Fassade Fliegengewicht Kompositum Proxy	Befehl Beobachter Besucher Iterator Memento Strategie Vermittler Zustand Zuständigkeitskette

Variation von Entwurfsaspekten

Aufgabe	Entwurfsmuster	variierende Aspekte
Erzeugungs-muster	Abstrakte Fabrik	Familien von Produktobjekten
	Erbauer	Erzeugung eines zusammengesetzten Objekts
	Fabrikmethode	Unterklasse, von der ein Objekt erzeugt wird
	Prototyp	Klasse des zu erzeugenden Objekts
	Singleton	das einzige Exemplar seiner Klasse
Struktur-muster	Adapter	Schnittstelle zu einem Objekt
	Brücke	Implementierung eines Objekts
	Dekorierer	Zuständigkeiten eines Objekts ohne neue Unterklassen
	Fassade	Schnittstelle zu einem Subsystem
	Fliegengewicht	Speicherkosten eines Objekts
	Kompositum	Struktur und Zusammensetzung eines Objekts
	Proxy	Zugriff auf ein Objekt sowie seinen Ort

Variation von Entwurfsaspekten
(Forts.)

Aufgabe	**Entwurfsmuster**	**variierende Aspekte**
Verhaltens-muster	Befehl	Zeitpunkt und Art der Ausführung einer Anfrage
	Beobachter	Anzahl der von einem Objekt abhängigen Objekte sowie der Art, in der die Objekte aktualisiert werden
	Besucher	Operationen, die auf Objekte ohne Änderung ihrer Klasse angewendet werden können
	Interpreter	Grammatik und Interpretierung einer Sprache
	Iterator	Zugriff und Traversierung der Elemente eines Aggregats
	Memento	Umfang und Zeitpunkt der Speicherung primitiver Information eines Objekts außerhalb von ihm
	Schablonenmethode	Schritte eines Algorithmus
	Strategie	ein Algorithmus
	Vermittler	zusammenarbeitende Objekte sowie die Art ihrer Zusammenarbeit
	Zustand	Zustände eines Objekts
	Zuständigkeitskette	Objekte, die eine Anfrage bearbeiten können

Wie Muster Entwurfsprobleme lösen

- Finden passender Objekte (z.B. Darstellung von Algorithmen als Objekte)
- Bestimmen von Objektgranularität (Anzahl und Größe)
- Spezifizieren von Objektschnittstellen
- Spezifizieren von Objektimplementationen
  - Klassenvererbung vs. Schnittstellenvererbung
  - Programmiere auf eine Schnittstelle hin, nicht auf eine Implementation!

Wie Muster Entwurfsprobleme lösen
(Forts.)

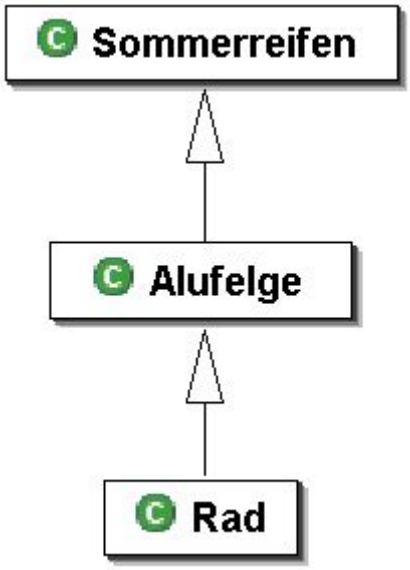
- Wiederverwendungsmechanismen anwenden
  - Vererbung (statisch) vs. Komposition (dynamisch)
  - Ziehe Objektkomposition der Klassenvererbung vor!

    2. Prinzip des wiederverwendbaren oo Entwurfs
  - Delegation
  - Vererbung vs. parametrisierbare Typen
- Strukturen der Lauf- und Übersetzungszeit aufeinander beziehen

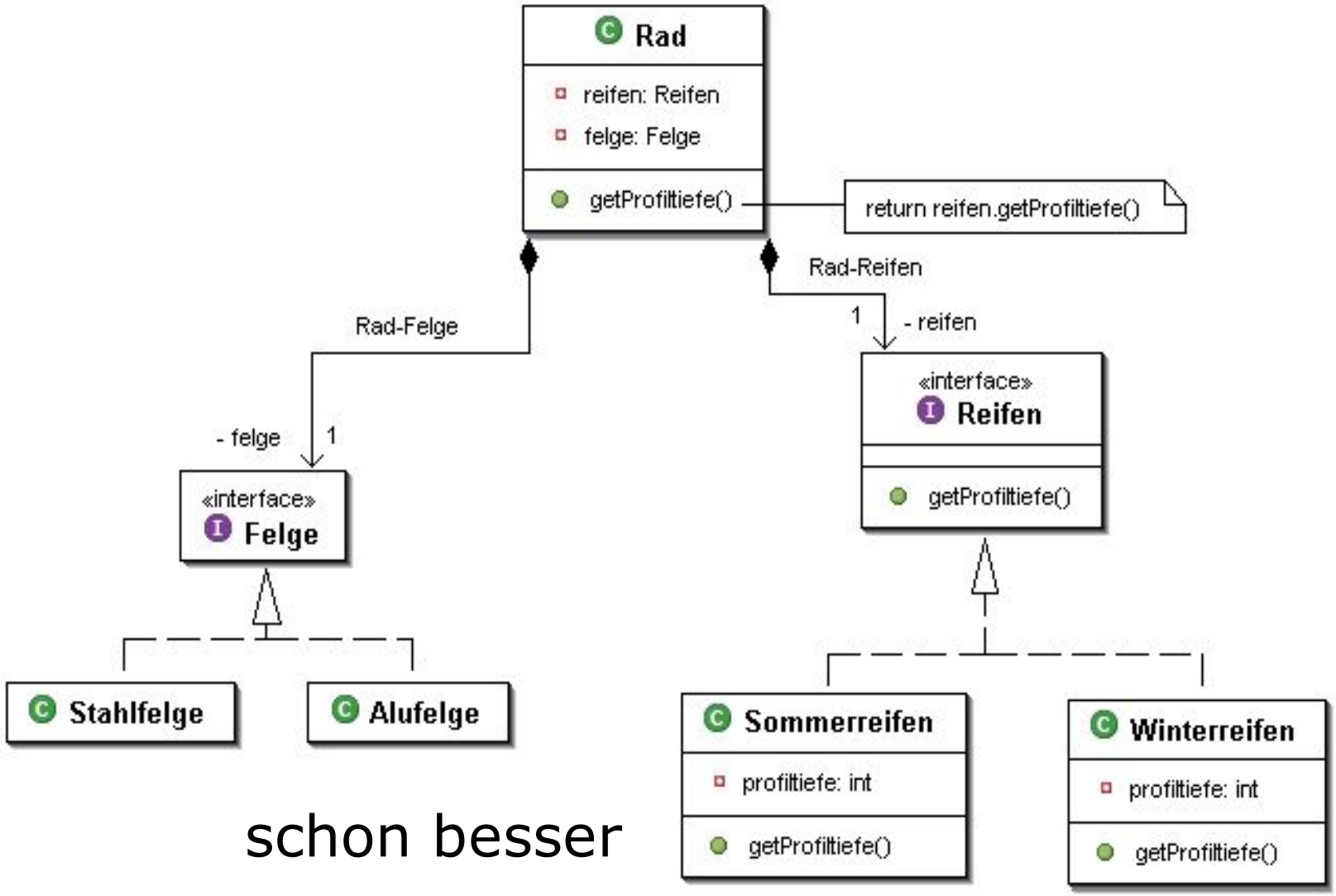
Wie Muster Entwurfsprobleme lösen
(Forts.)

- Veränderungen in Entwürfen vorhersehen
  - Erzeugen eines Objekts unter expliziter Nennung seiner Klasse
  - Abhängigkeit von spezifischen Operationen
  - Abhängigkeit von Hard- und Softwareplattformen
  - Abhängigkeit von Objektrepräsentationen oder Objektimplementationen
  - Algorithmische Abhängigkeiten
  - Enge Kopplung
  - Funktionalitätserweiterung durch Unterklassenbildung
  - Unmöglichkeit, Klassen bequem zu ändern

Prinzipien des oo Entwurfs



so nicht!



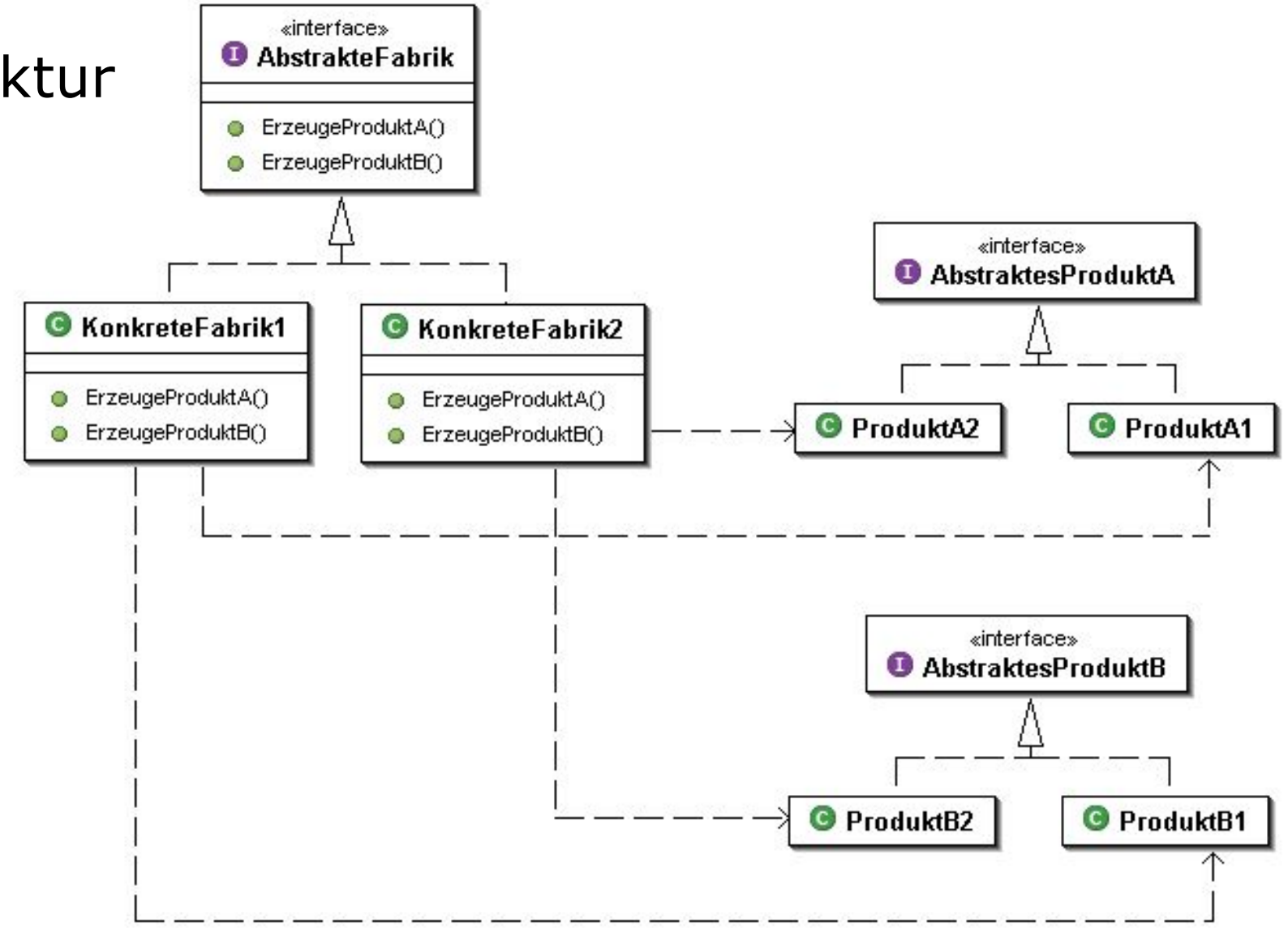
schon besser

Abstrakte Fabrik

- Zweck
- auch bekannt als
  - Kit
- Motivation
- Anwendbarkeit

Abstrakte Fabrik (Forts.)

- Struktur



Abstrakte Fabrik (Forts.)

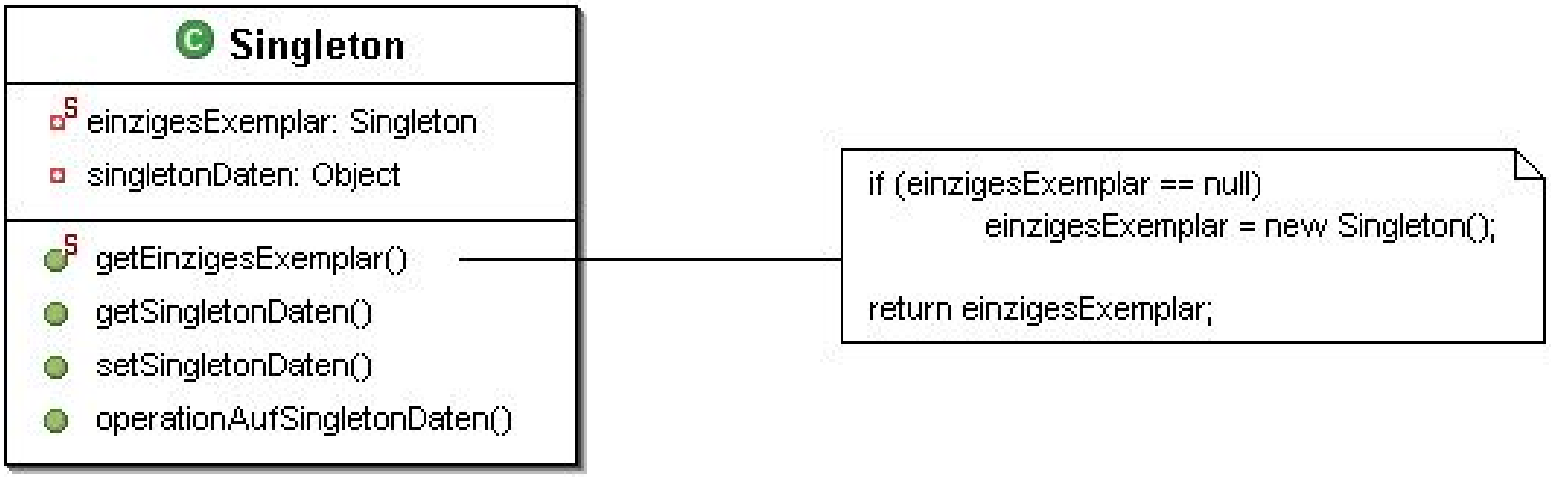
- Konsequenzen
- Implementierung
- Beispiel
- bekannte Verwendungen
- verwandte Muster

Singleton

- Zweck
- Motivation
- Anwendbarkeit

Singleton (Forts.)

- Struktur



Singleton (Forts.)

- Konsequenzen
- Implementierung
- Beispiel
- bekannte Verwendungen
- verwandte Muster

Adapter

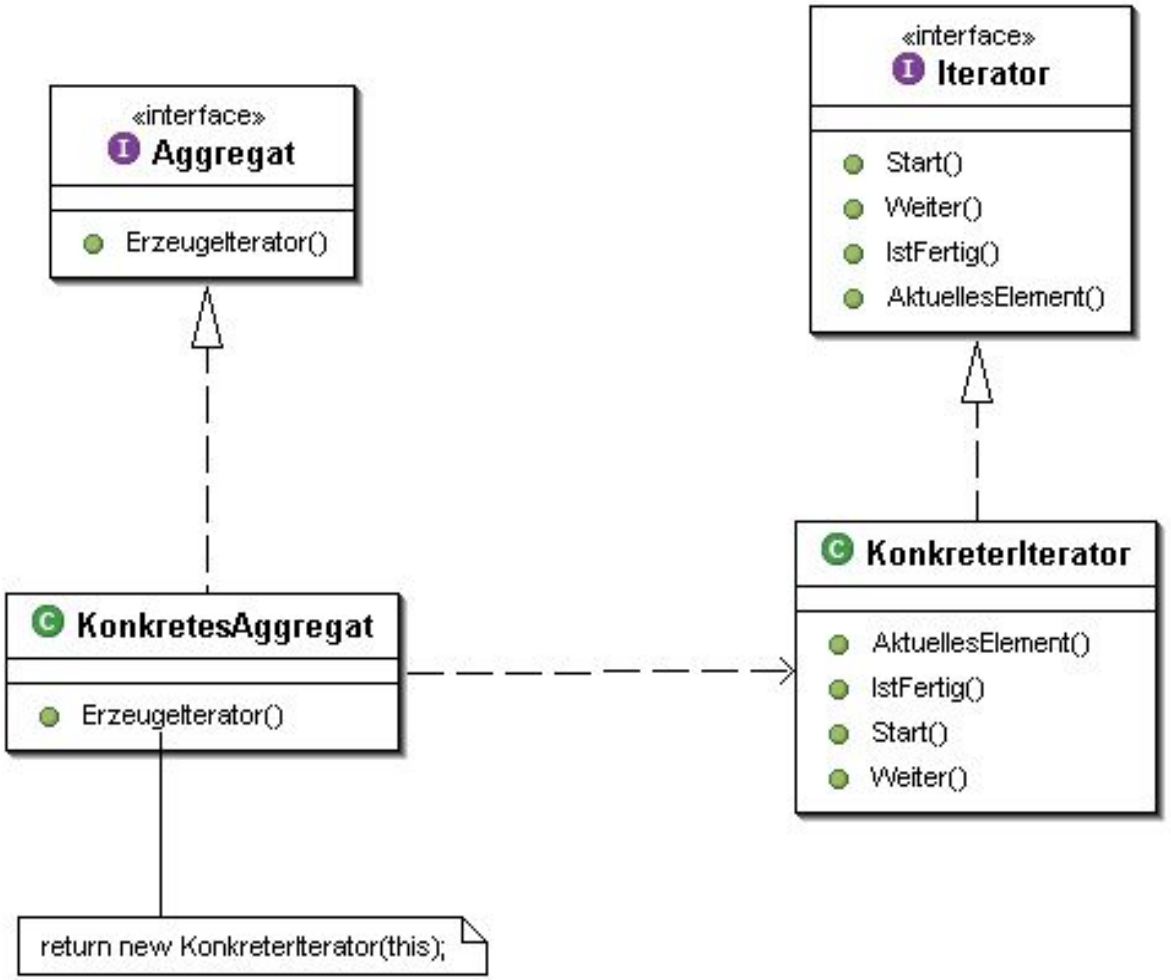
- ???

Iterator

- Zweck
- auch bekannt als
  - Cursor
- Motivation
- Anwendbarkeit

Iterator (Forts.)

- Struktur



Iterator (Forts.)

- Konsequenzen
- Implementierung
- Beispiel
- bekannte Verwendungen
- verwandte Muster

Fazit

- bessere Wiederverwendbarkeit, mehr Flexibilität
- gemeinsames Entwurfsvokabular
- Dokumentations- und Lernhilfe
- Ergänzung zu existierenden Methoden
- Ziel für Refaktorisierungen

Fazit (Forts.)

„Muster sind in jeder gut strukturierten objektorientierten Architektur zu finden. Ich gehen sogar soweit, daß ich die Qualität eines objektorientierten Systems nach der Sorgfältigkeit bewerte, welche die Entwickler der Zusammenarbeit zwischen den Objekten gewidmet haben.“

Grady Booch, aus Geleitwort „Entwurfsmuster“

Quellen

- E. Gamma, R. Helm, R. Ohnson, J. Vlissides: Entwurfsmuster, Addison-Wesley 2001
- Zdenko Denny Vrandecic, Entwurfsmuster (Design Patterns), Uni Stuttgart
- Brad Appleton, Patterns and Software: Essential Concepts and Terminology