Zusammenfassung Entwurfsmuster SWT

Jopunkt

2012

Inhaltsverzeichnis

1 Allgemein

Entwurfsmuster (en. Desing Patterns) sind bewehrte Lösungsschablonen für wiederkehrende Entwurfsprobleme. Sie stellen damit eine wiederverwendbare Vorlage zur Problemlösung dar, die in einem bestimmten Zusammenhang einsetzbar ist.

1.1 Vorteile

. . . oder guter Zwecke, Sinn, etc.:

- Ein Entwurfsmuster beschreibt eine Familie von Lösungen für ein Software-Entwurfsproblem.
- Entwurfsmuster haben eine Widerverwendbarkeit von Entwurfswissen erschaffen
- Entwurfsmuster sind für den Entwurf (oder das Programmieren im Großen) was Algorithmen für das Programmieren im kleinen sind
- Entwurfsmuster verbessern die Kommunikation im Team (da sie als Terminologie für abstraktes dienen)
- Entwurfsmuster erfassen wesentliche Konzepte und bringen sie eine verständliche Form, fördern so den Stand der Kunst
  - helfen Entwürfe zu verstehen
  - dokumentieren Entwürfe kurz und knapp
  - verhindern unerwünschten Architektur-Drift
  - vermeiden die Neuerfindung des Rades
  - helfen weniger erfahren
- Entwurfsmuster können Code-Qualität und Code-Struktur verbessern

1.2 Nachteile

Der erfolgreiche Einsatz von Entwurfsmustern in der Vergangenheit kann dazu verleiten, die Entwurfsmuster als Wunderwaffe und Garant für gutes Design anzusehen. Unerfahrene Entwickler können geneigt sein, möglichst viele bekannte Muster zu verwenden und dabei übersehen, dass in ihrem Fall vielleicht eine elegantere Lösung ohne den Einsatz von Mustern möglich wäre. Entwurfsmuster garantieren nicht, dass der Entwurf gut ist. Insofern ist die Anwendung zu vieler oder ungeeigneter Entwurfsmuster ein Antimuster.

2 Übersicht

Entkopplungsm.	Variantenm.	Zustandsm.	Steuerungsm.	Bequeml.keitsm.
Adapter	Abstrakte Fabrik	Einzelstück	Befehl	Beq-Klasse
Beobachter	Besucher	Fliegengewicht	Master-Worker	Beq-Methode
Brücke	Fabrikmethode	Memento		Fassade
Iterator	Kompositum	Prototyp		Null-Objekt
Stellvertreter	Schablonenmethode	Zustand		
Dekorierer	Strategie			
Vermittler				
7	6	5	2	4

Insgesamt 24 Entwurfsmuster.

3 Entkopplungsmuster

- Entkopplungsmuster teilen ein System in mehrere Einheiten, so dass einzelne Einheiten unabhängig voneinander erstellt, verändert, ausgetauscht und wiederverwendet werden können.
- Der Vorteil von Entkopplungsmuster ist, dass ein System durch lokale Änderungen verbessert, angepasst und erweiter werden kann, ohne das ganze System zu modifizieren.
- Mehrere Entkopplungsmuster enthalten ein Kopplungsglied, das entkoppelte Einheiten über eine Schnittstelle kommunizieren lässt. Diese Kopplungsglieder sind auch für das Koppeln unabhängig erstellter Einheiten brauchbar.

Die Entkopplungsmuster:

- *Adapter:* Passt die Schnittstelle einer Klasse an eine andere von ihren Klienten erwartete Schnittstelle an. Überbrückt inkompatible Schnitstellen.
- *Beobachter (Observer):* Definiert eine 1-zu-n Abhängigkeit zwischen Objekten, so dass Änderungen eines Zustandes eines Objektes dazu führt, dass alle abhängigen Objekte benachrichtigt und automatisch aktualisiert werden.
- *Brücke:* Entkopplt eine Abstraktion von ihrer Implementierung, so dass beide unabhängig voneinander variiert werden können.
- *Iterator:* Ermöglicht den sequentillen Zugriff auf die Elemente eines zusammengesetzten Objekts, ohne seine zugrundeliegende Repräsentation offenzulegen.
- *Stellvertreter:* Kontrolliere den Zugriff auf ein Objekt mit Hilfe eines vorgelagerten Stellvertreterobjekts.
- *Dekorierer:* Fügt dynamisch neue Funktionalitäten zu einem Objekt hinzu.
- *Vermittler:* Definiert ein Objekt, welches das Zusammenspiel einer Menge in sich kapselt. Vermittler fördern lose Kopplung, indem sie Objekte abhalten, aufeinander explizit Bezug zu nehmen. Sie ermöglichen es, das Zusammenspeil der Objekte unabhängig zu variieren.

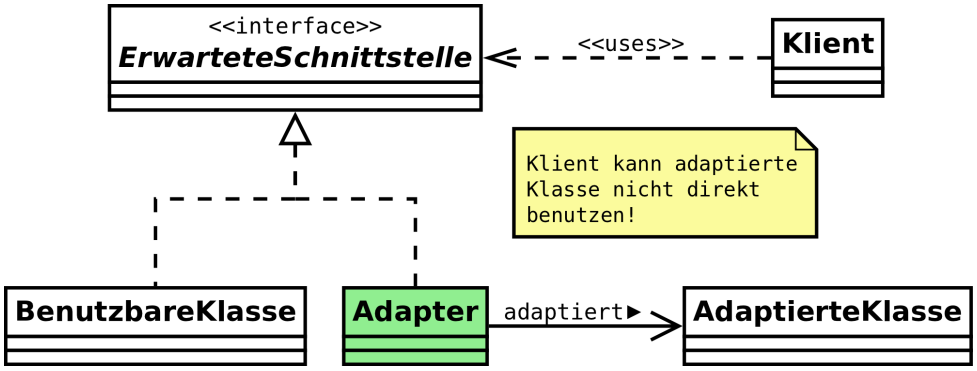
3.1 Adapter

oder auch Wandler, Umwickler, wrapper.

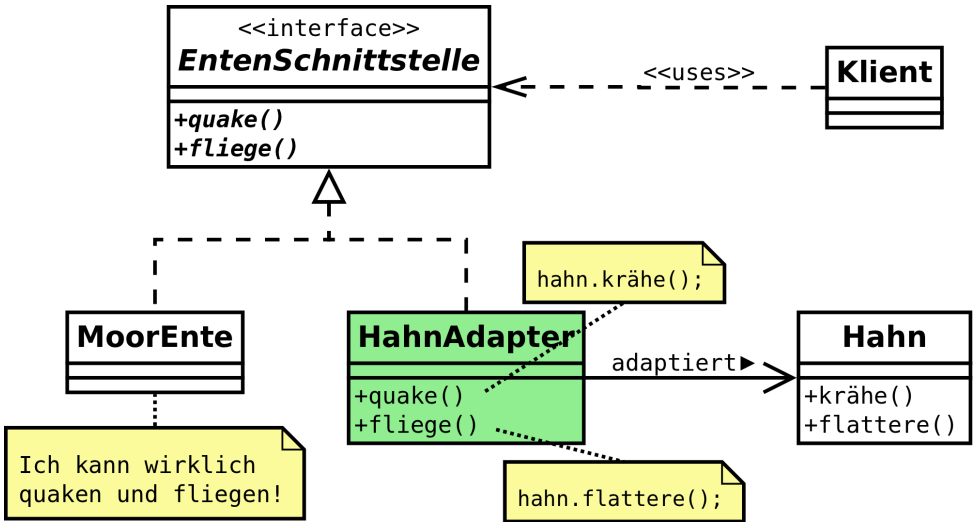
> Passt die Schnittstelle einer Klasse an eine andere von ihren Klienten erwartete Schnittstelle an. Überbrückt inkompatible Schnitstellen.

Das Adaptermuster eignet sich zur Verwendung, wenn man eine ex. Klasse verwenden möchte, deren Schnittstelle inkompatibel ist oder wenn man eine wiederverwendbare Klasse erstellen will, die mit anderen auch unbekannten Klassen arbeiten können soll.

Allgemeine Struktur



Beispiel



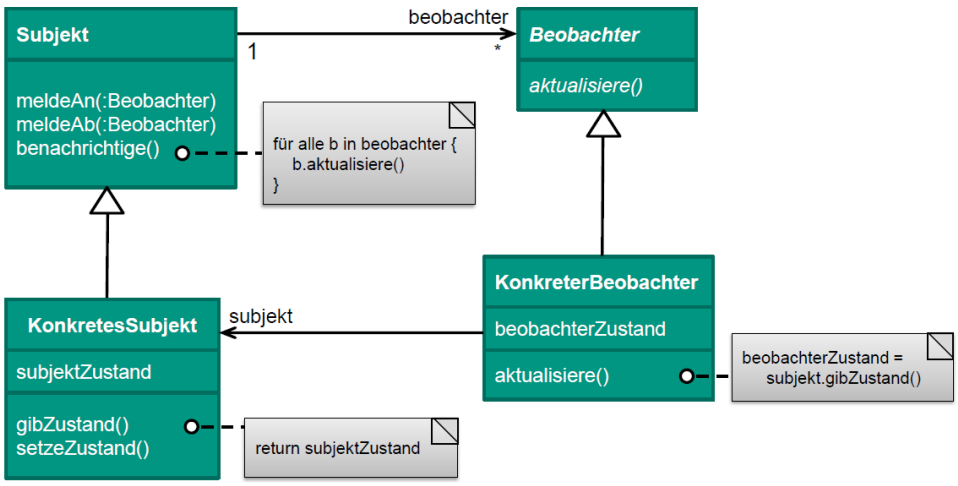
3.2 Beobachter

oder auch observer, publisher-subscriber, Abhängigkeit.

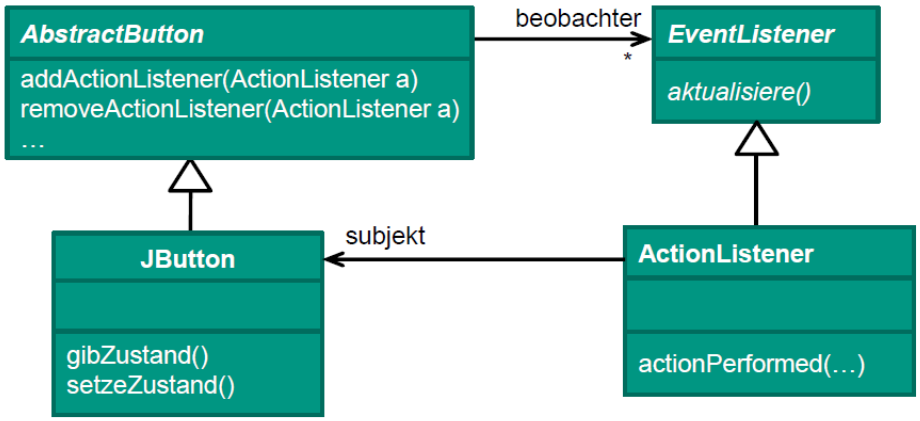
> Definiert eine 1-zu-n Abhängigkeit zwischen Objekten, so dass Änderungen eines Zustandes eines Objektes dazu führt, dass alle abhängigen Objekte benachrichtigt und automatisch aktualisiert werden.

- Anwendbar, wenn die Änderung eines Objekts die Änderung anderer Objekte verlangt und man nicht weiß, wie viele und welche Objekte geändert werden müssen.
- Wenn ein Objekt andere benachrichten muss, ohne Annahmen über diese Objekte zu treffen.
- Wenn eine Abstraktion zwei Aspekte besitzt, wobei einer von dem anderen abhängt. Die Kapselung dieser Aspekte in separaten Objekten ermöglicht unabhängige Wiederverwendbarkeit.

Allgemeine Struktur



Beispiel



3.3 Brücke

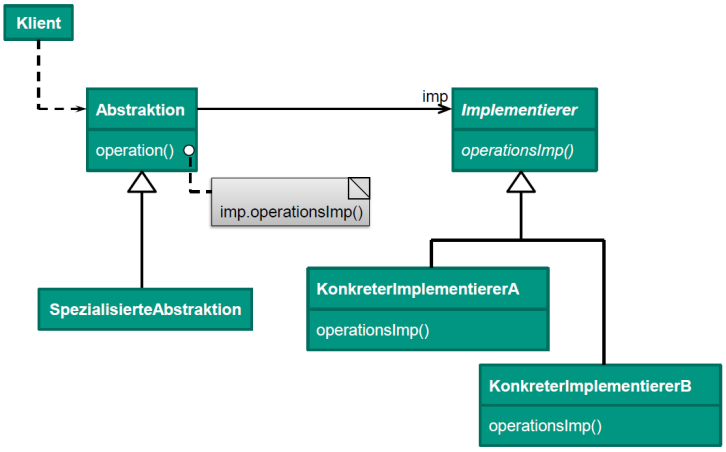
oder auch bridge, handle, body.

> Entkopplt eine Abstraktion von ihrer Implementierung, so dass beide unabhängig voneinander variiert werden können.

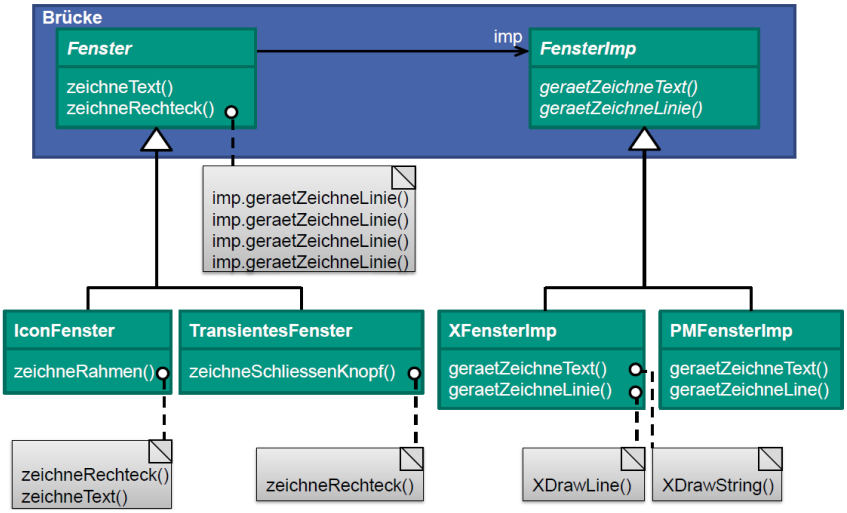
Die Vorteile einer Brücke bestehen darin, dass Abstraktion und Implementierung entkoppelt werden. Die Implementierung ist weiterhin während der Laufzeit dynamisch änderbar und die Erweiterbarkeit von Abstraktion und Implementierung wird verbessert.

Durch Angabe eines Parameters bei der Erzeugung einer Abstraktion kann die Implementierung gewählt werden, zudem wird die Implementierung für den Klienten vollständig versteckt. Eine starke Vergrößerung der Anzahl der Klassen kann vermieden werden.

Allgemeine Struktur



Beispiel



Eine Brücke kann verwendet werden, wenn eine dauerhafte Verbindung zwischen Abstraktion und Implementierung vermieden werden soll. Wenn sowohl Abstraktion als auch Implementierung durch Unterklassenbildung erweiterbar sein soll. Wenn Änderungen in der Implementierung einer Abstraktion keine Auswirkung auf Klienten haben soll.

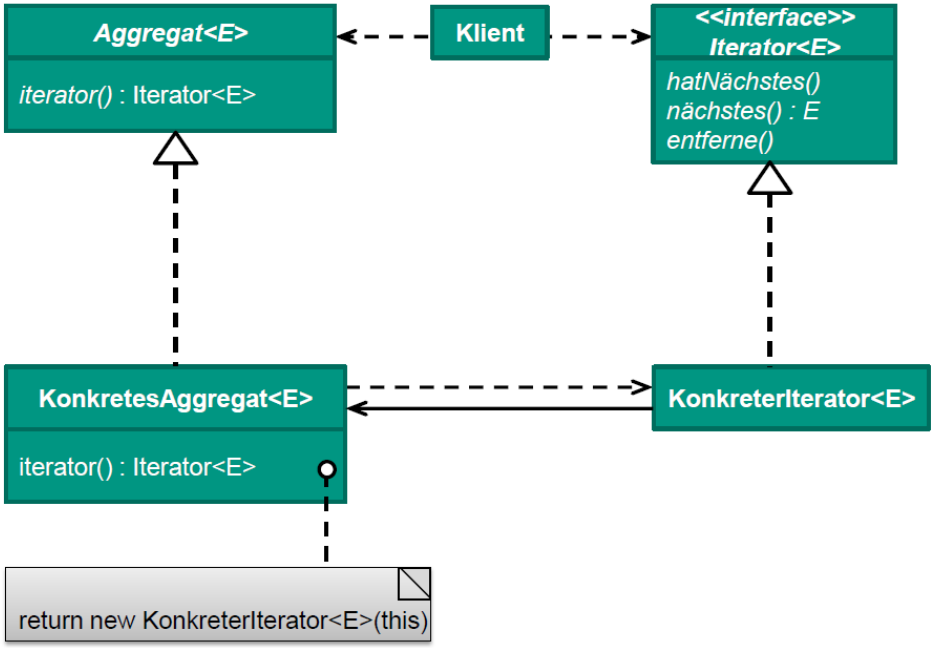
3.4 Iterator

oder auch Enumerator.

> Ermöglicht den sequentillen Zugriff auf die Elemente eines zusammengesetzten Objekts, ohne seine zugrundeliegende Repräsentation offenzulegen.

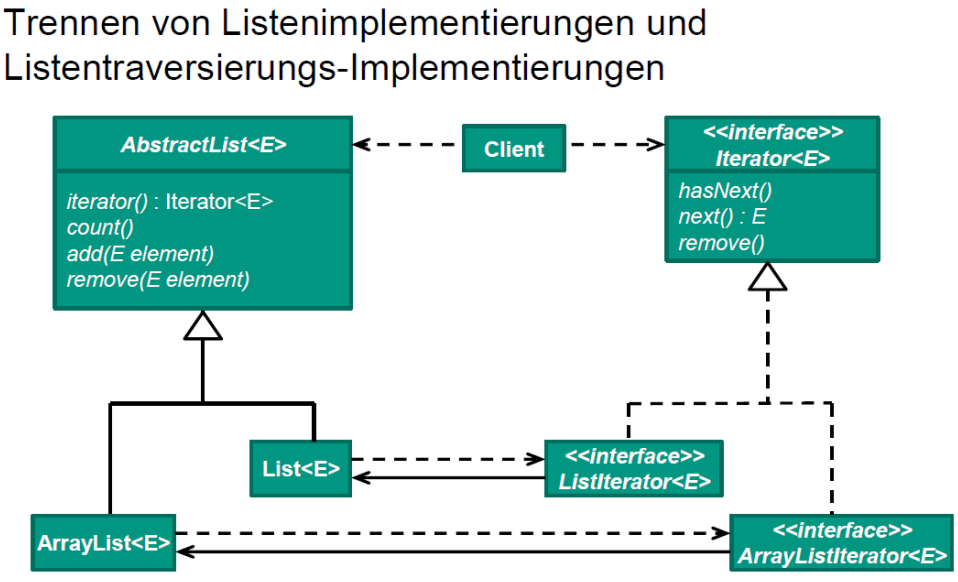
Ein Iterator kann genutzt werden, um einen gekapselten Zugriff auf Inhalte eines Objekts zu ermöglichen oder eine einheitliche Schnittstelle zur Traversierung zu bieten.

Allgemeine Struktur



Beispiel

Wird z.B. in allen Java-Collections verwendet und mit dem Interface Iterable für foreach Schleifen oder Iterator für anderes verwendbar gemacht.



3.5 Stellvertreter

oder auch proxy, Surrogat, surrogate.

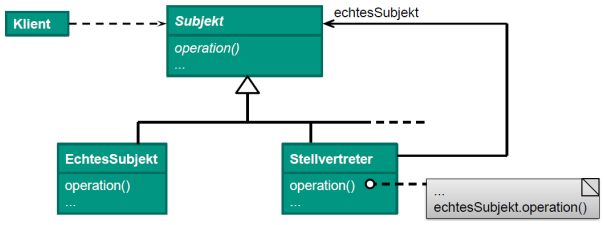
Kontrolliere den Zugriff auf ein Objekt mit Hilfe eines vorgelagerten Stellvertreterobjekts.

Funktionen des Stellvertreters

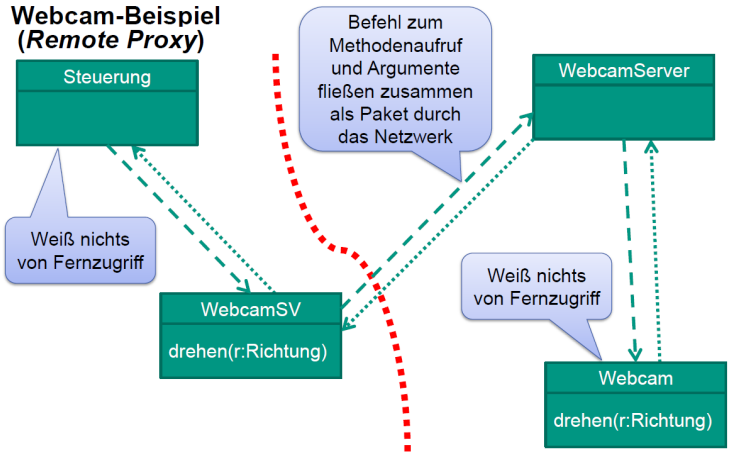
Anwendbar ist das Stellvertretermuster, sobald es den Bedarf nach einer anpassungsfähigen und intelligenteren Referenz auf ein Objekt als einfachen Zeiger gibt.

- Platzhalter (virtual proxy) erzeugt teure Objekte auf Verlangen.
- Schutzwand (firewall) kontrolliert den Zugriff auf das Orignalobjekt z.B. mit verschiedenen Zugriffsrechten.
- Synchornisierungsvertreter (synchronization proxy) koordiniert den Zugriff mehrerer Fäden auf ein Objekt.
- Puffernder Stellvertreter (caching proxy) lädt persistents Objekt erst dann, wenn es das erste Mal angesprochen wird.
- Fernzugriffsvertreter (remote proxy) stellt einen lokalen Stellvertreter für ein Objekt in einem anderen Adressraum dar.
- Protokollierender Stellvertreter zählt Referenzen auf das egtl. Objekt, so dass es automatisch freigegeben werden kann, oder andere Zugriffsinformationen.

Allgemeine Struktur



Beispiel



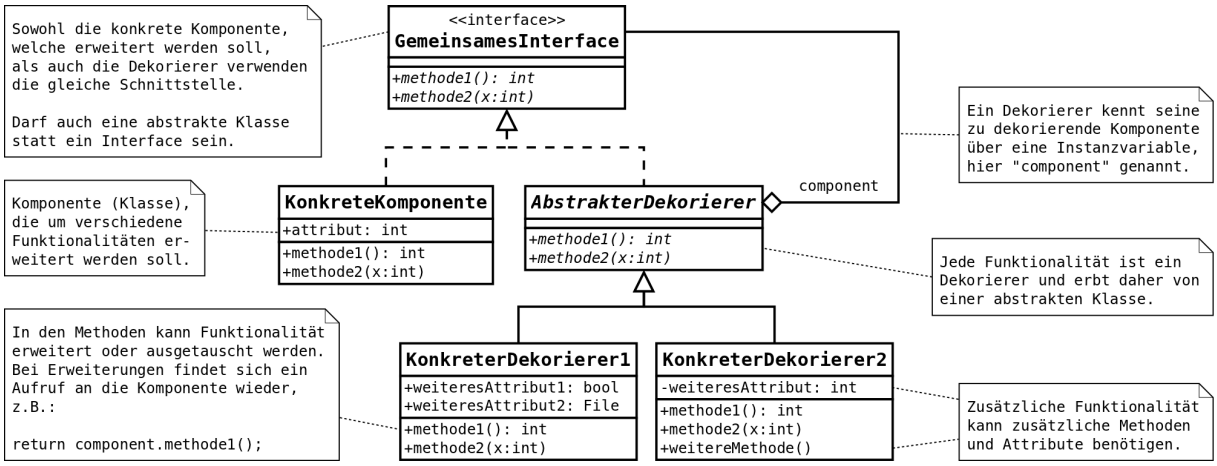
3.6 Dekorierer

oder auch decorator.

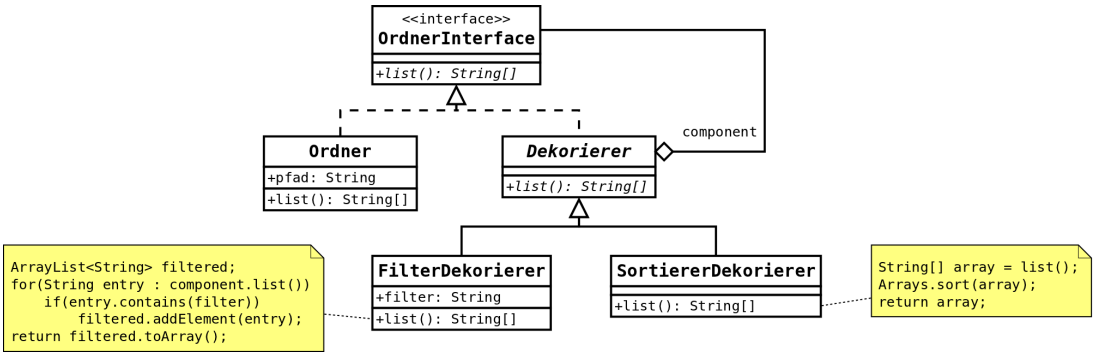
Fügt dynamisch neue Funktionalitäten zu einem Objekt hinzu.

Kann alternativ zu Vererbung verwendet werden.

Allgemeine Struktur



Beispiel



Vergleich zum Stellvertreter

Dekorierer fügt Objektfunktionalität hinzu, ohne Objekt zu ändern. Kann dadurch auch die Subjektschnittstelle erweitern.

Stellvertreter ist ein Zugriffssteuerung, kann genauso wie das Subjekt selbst verwendet werden, kann Latenz und Methoden verstecken. Ist ein eigenes Objekt mit Subjekt „im Hintergrund“.

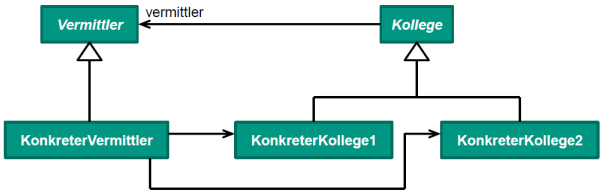
3.7 Vermittler

oder auch mediator.

> Definiert ein Objekt, welches das Zusammenspiel einer Menge in sich kapselt. Vermittler fördern lose Kopplung, indem sie Objekte abhalten, aufeinander explizit Bezug zu nehmen. Sie ermöglichen es, das Zusammenspeil der Objekte unabhängig zu variieren.

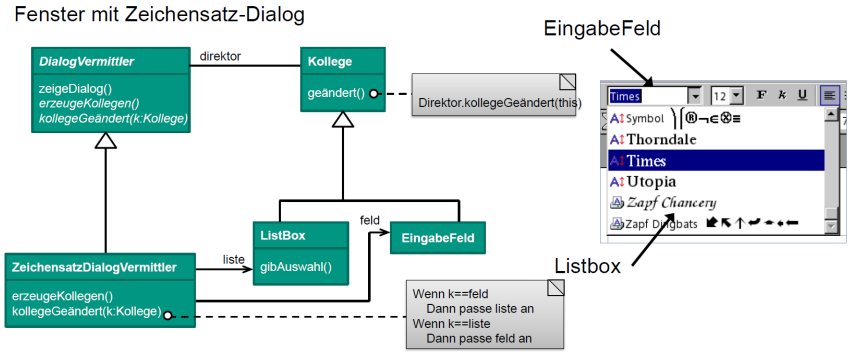
Anwendbar, wenn eine Menge von Objekten vorliegt, die in wohldef., aber komplexer Weise zusammenarbeiten. Abhängigkeiten sonst schwer zu strukturieren und zu verstehen. Wenn ein auf mehrere Klassen verteiltes Verhalten maßgeschneidert werden soll, ohne viele Unterklassen bilden zu müssen.

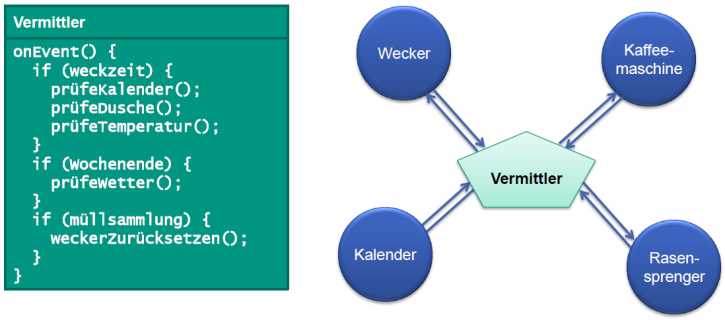
Allgemeine Struktur



Beispiel

Zum Beispiel Abhängigkeiten zwischen Elementen einer Dialogbox. So muss z.B. ein Knopf deaktiviert sein, wenn ein bestimmtes Texteingabefeld leer ist. Jedoch besitzen unterschiedliche Dialogboxen unterschiedliche Abhängigkeiten zwischen Elementen. Hierbei ist eine individuelle Anpassung in Unterklassen zu mühsam und schlecht wiederverwendbar. Das Gesamtverhalten wird deshalb in einem Vermittlerobjekt gekapselt.





4 Varianten-Muster

- In Mustern dieser Gruppe werden Gemeinsamkeiten von verwandten Einheiten aus ihnen herausgezogen und an einer einzigen Stelle beschrieben.
- Aufgrund ihrer Gemeinsamkeiten können unterschiedliche Komponenten im gleichen Programm danach einheitlich verwendet werden, und Code-Redudanz vermieden werden.

Die Varianten-Muster:

- *Abstrakte Fabrik:* Bietet eine Schnittstelle zum Erzeugen von Familien verwandter oder voneinander abhängiger Objekte, ohne ihre konkreten Klassen zu bennen.
- *Besucher:* Kapselt eine auf den Elementen einer Objektstruktur auszuführende Operation als ein Objekt. Ermöglicht es, eine neue Operation zu definieren, ohne die Klassen der von ihr berarbeiteten Elemente zu verändern.
- *Fabrikmethode:* Definiert eine Klassenschnittstelle mit Operationen zum Erzeugen eines Objekts, aber lässt Unterklassen entscheiden, von welcher Klasse das zu erzeugende Objekt ist. Ermöglicht einer Klasse, die Erzeugung von Objekten an Unterklassen zu delegieren.
- *Kompositum:* Fügt Objekte zu Baumstrukten zusammen, um Bestandshierarchien zu repräsentieren. Das Muster ermöglicht es Klienten, sowohl als einzelne Objekte, als auch Aggreagate einheitlich zu behandeln.
- *Schablonenmethode:* Definiert das Skelett eines Algorithmus in einer Operation und delegiert einzelne Schritte an Unterklassen. Ermöglicht es Unterklassen, bestimmte Schritte eines Algorithmus zu überschreiben, ohne seine Struktur zu verändern.
- *Strategie:* Definiert eine Familie von Algorithmen, kapselt sie und macht sie austauschbar. Ermöglicht es, den Algorithmus unabhängig von nutzenden Klienten zu variieren.

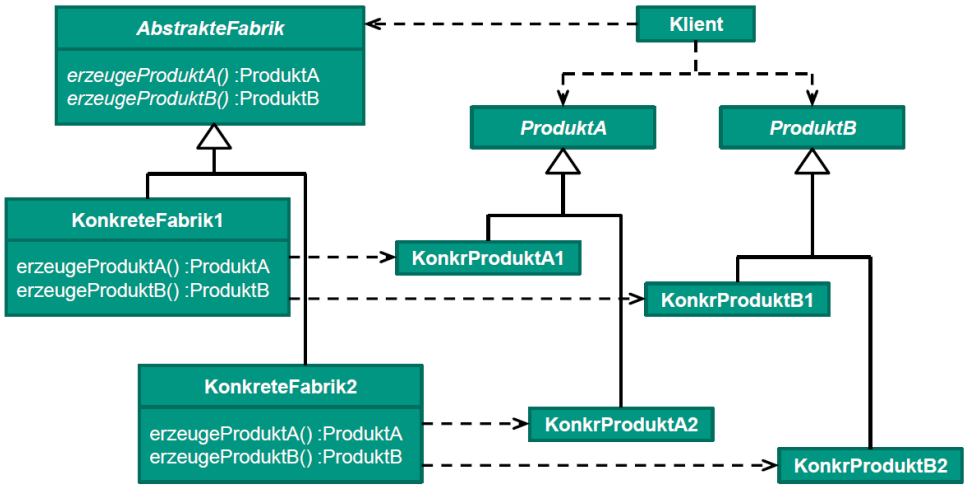
4.1 Abstrakte Fabrik

oder auch abstract factory, Kit.

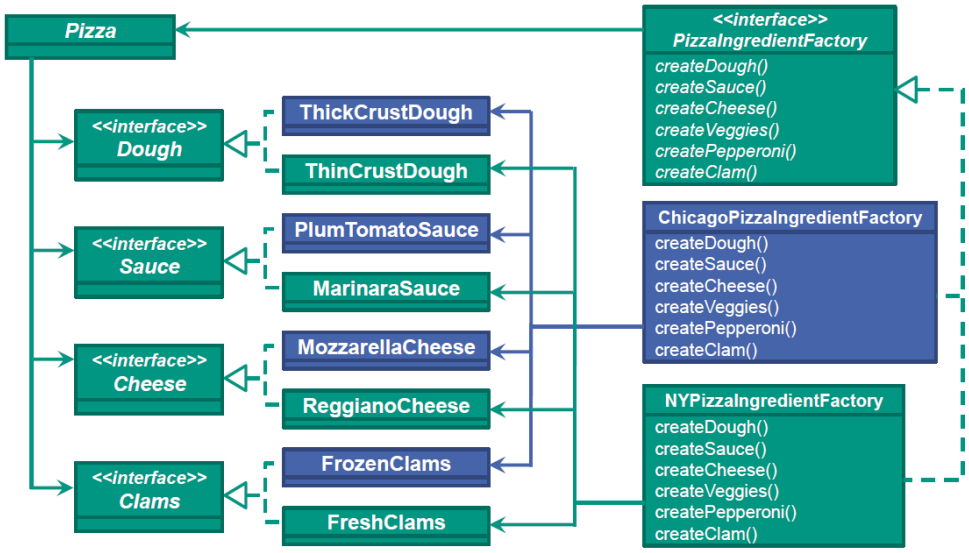
Bietet eine Schnittstelle zum Erzeugen von Familien verwandter oder voneinander abhängiger Objekte, ohne ihre konkreten Klassen zu benennen.

- Anwendbar, wenn ein System unabhängig davon sein soll, wie seine Produkte erzeugt, zusammengesetzt und repräsentiert werden.
- Wenn ein System mit einer von mehreren Produktfamilien konfiguert werden soll.
- Wenn eine Familie von aufeinander abgestimmten Produktobjekten zusammen verwendet werden soll, und dies erzwungen werden muss.

Allgemeine Struktur



Beispiel



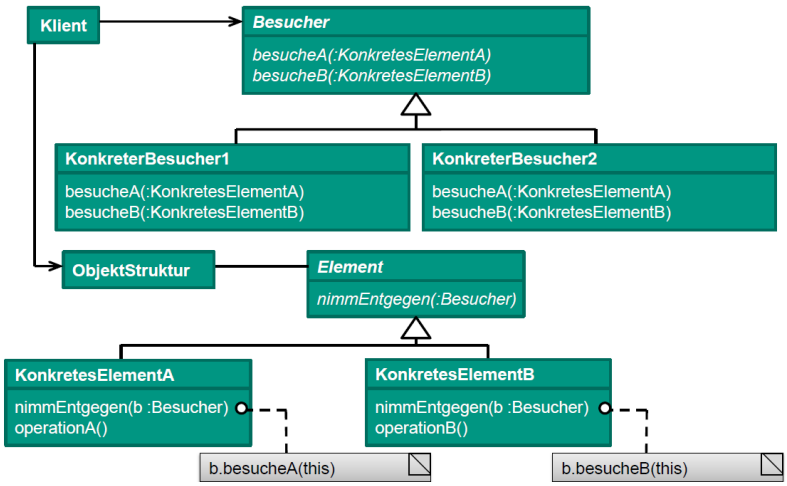
4.2 Besucher

oder auch visitor.

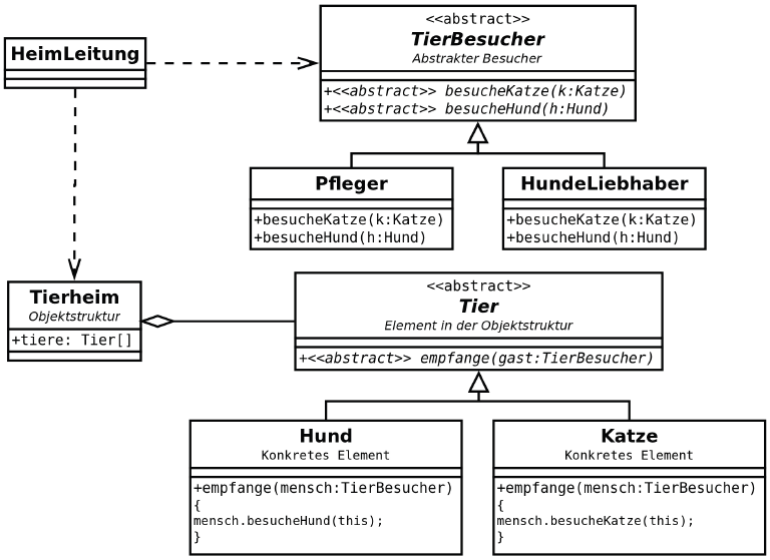
> Kapselt eine auf den Elementen einer Objektstruktur auszuführende Operation als ein Objekt. Ermöglicht es, eine neue Operation zu definieren, ohne die Klassen der von ihr berarbeiteten Elemente zu verändern.

- Das Besucher-Muster kann helfen, wenn eine Objektstruktur viele Klassen von Objekten mit unterschiedlichen Schnittstellen enthält und Operationen auf diesen Objekten ausgeführt werden sollen, die von ihren konkreten Klassen abhängen.
- Wenn mehrere unterschiedliche und nicht miteinander verwandete Operationen auf den Objekten einer Objektstruktur ausgeführt werden müssen und diese Klassen nicht mit diesen Operationen „verschmutzt“ werden sollen.
- Wenn sich die Klassen praktisch nie ändern, aber häuft neue Operationen auf deren Objektstruktur definiert werden.

Allgemeine Struktur



Beispiel



4.3 Fabrikmethode

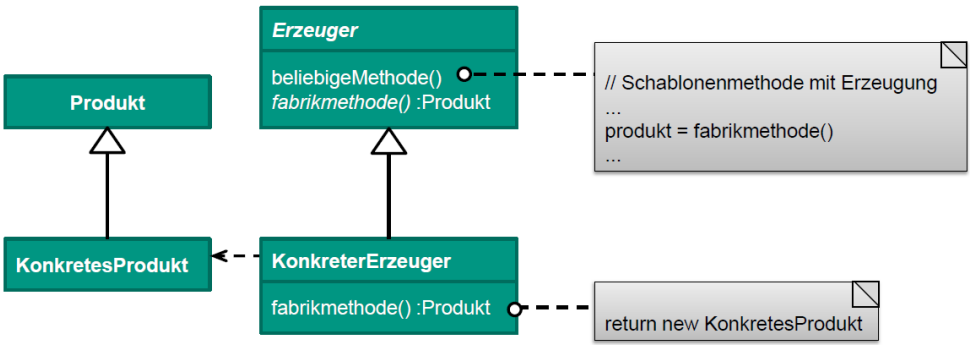
oder auch factory method.

> Definiert eine Klassenschnittstelle mit Operationen zum Erzeugen eines Objekts, aber lässt Unterklassen entscheiden, von welcher Klasse das zu erzeugende Objekt ist. Ermöglicht einer Klasse, die Erzeugung von Objekten an Unterklassen zu delegieren.

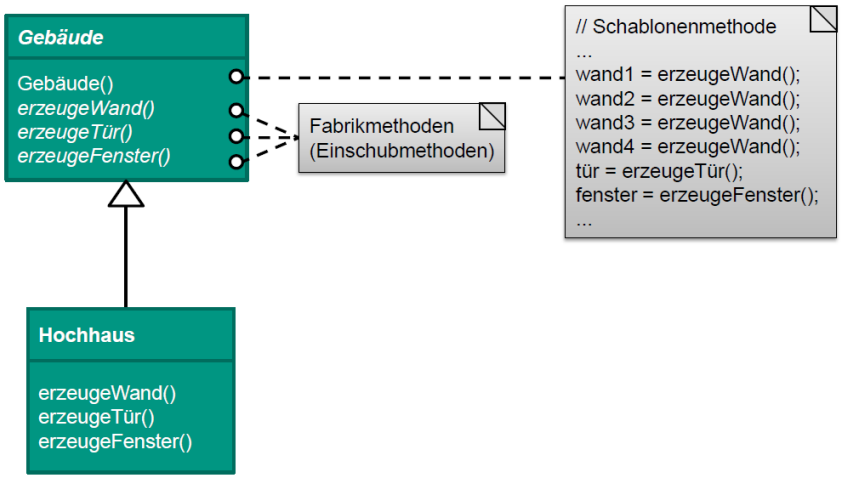
- Anwendbar, wenn eine Klasse von Objekten, die sie erzeugen muss, nicht im voraus kennen kann.
- Wenn eine Klasse mächte, dass ihre Unterklassen die von ihr zu erzeugenden Objekte festlegen.
- Wenn Klassen Zuständigkeiten an eine von mehreren Hilsunterklassen delegieren sollen und das Wissen, an welche Hilfsunterklasse die Zuständigkeit delegiert wird, lokalisiert werden soll.

Eine Fabrikmethode ist die Einschubmethode bei einer Schablonenmethode für Objekterzeugung.

Allgemeine Struktur



Beispiel



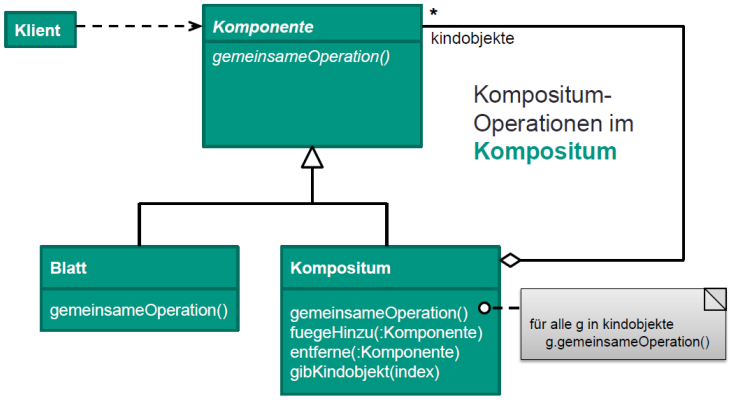
4.4 Kompositum

oder auch composite, whole-part, Bestandshierarchie.

> Fügt Objekte zu Baumstrukten zusammen, um Bestandshierarchien zu repräsentieren. Das Muster ermöglicht es Klienten, sowohl als einzelne Objekte, als auch Aggreagate einheitlich zu behandeln.

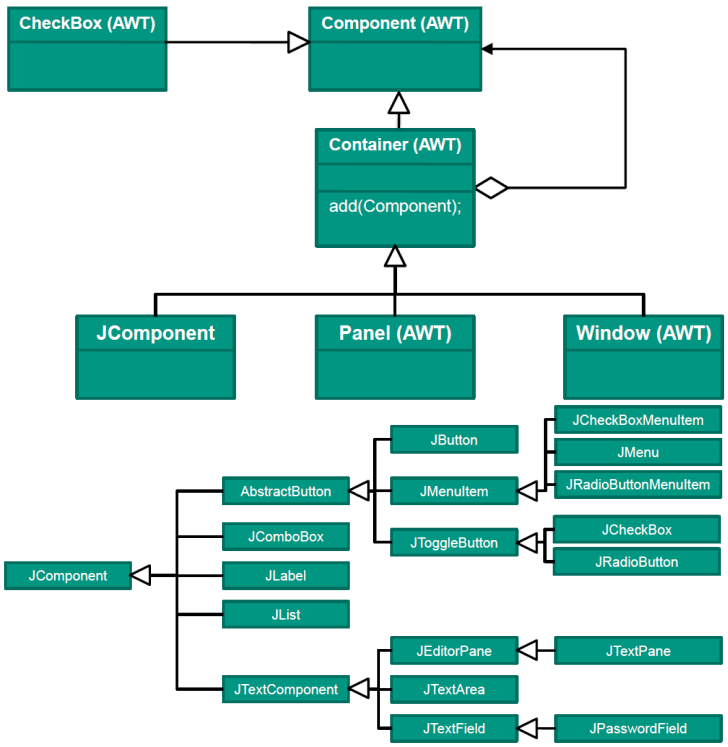
Anwendbar, bei Bestands-Hierarchien oder wenn die Klienten nicht zwischen zusammengesetzen und einzelnen Objekten unterscheiden sollen müssen.

Allgemeine Struktur



Beispiel

Javas AWT-Klassen sind nach dem Kompositum-Muster gebaut. Da alle von Container erben, können sie jeweils selbst wieder Elemente aufnehmen.



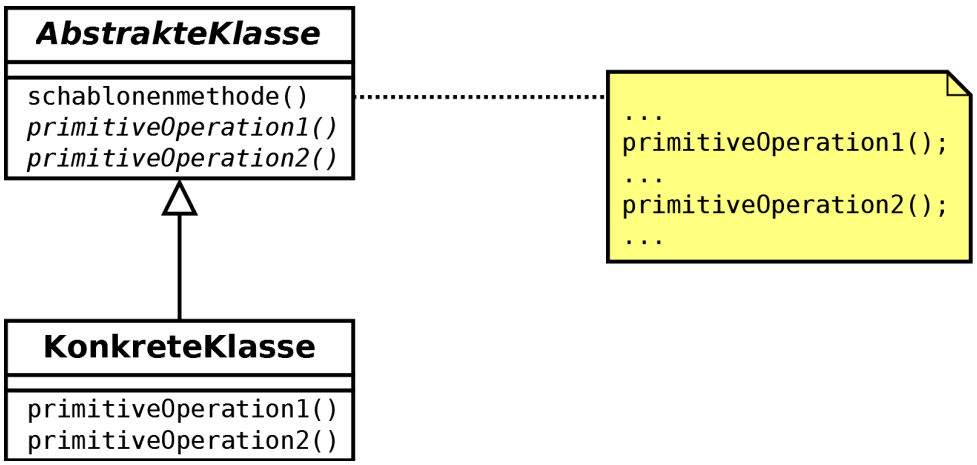
4.5 Schablonenmethode

oder auch .

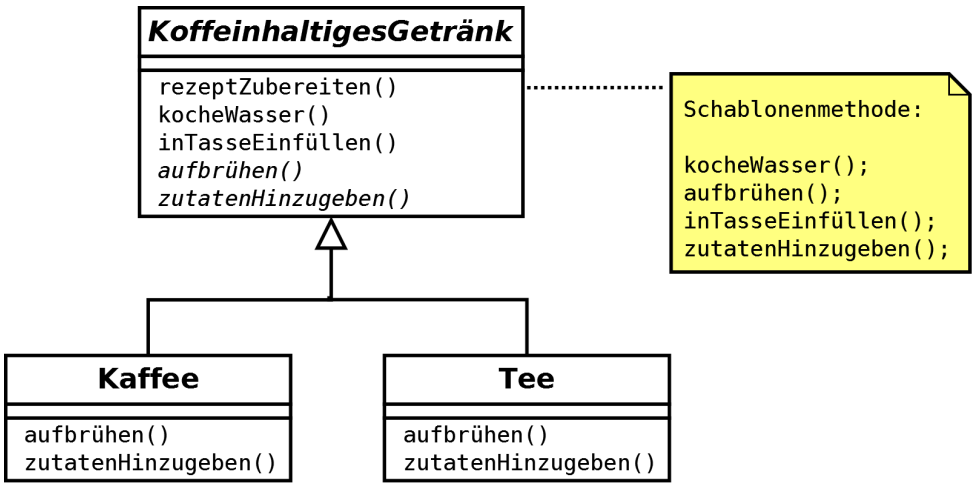
> Definiert das Skelett eines Algorithmus in einer Operation und delegiert einzelne Schritte an Unterklassen. Ermöglicht es Unterklassen, bestimmte Schritte eines Algorithmus zu überschreiben, ohne seine Struktur zu verändern.

- Findet Anwendung, um die invarianten Teile eines Algorithmus genau einmal festzulegen und es dann Unterklassen zu überlassen, das variierende Verhalten zu implementieren.
- Wenn gemeinsame Verhalten aus Unterklassen herausfaktorisiert und in einer allgemeinen Klasse platziert werden soll.
- Um die Erweiterung durch Unterklassen zu kontrollieren. Eine Schablonenmethode lässt sich so definieren, dass sie Einschubmethoden an bestimtmen Stellen aufruft und damit Erweiterungen nur an diesen Stellen zulässt.
- Häufig bei Rahmenarchitekturen genutzt.

Allgemeine Struktur



Beispiel



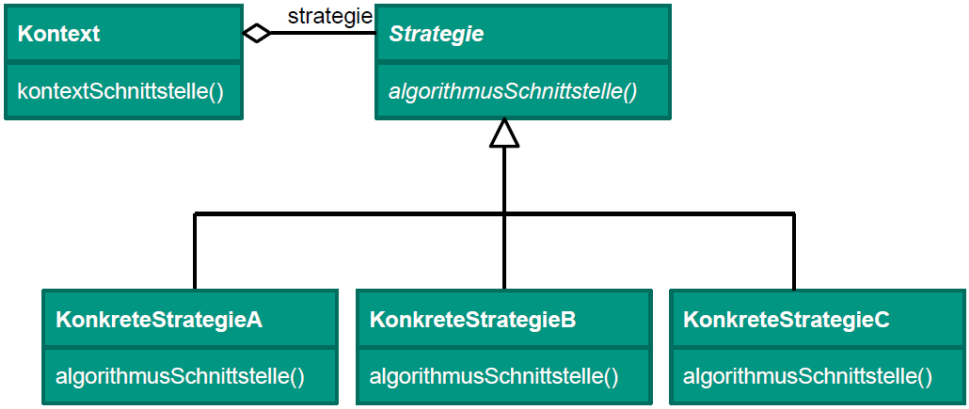
4.6 Strategie

oder auch strategy, policy.

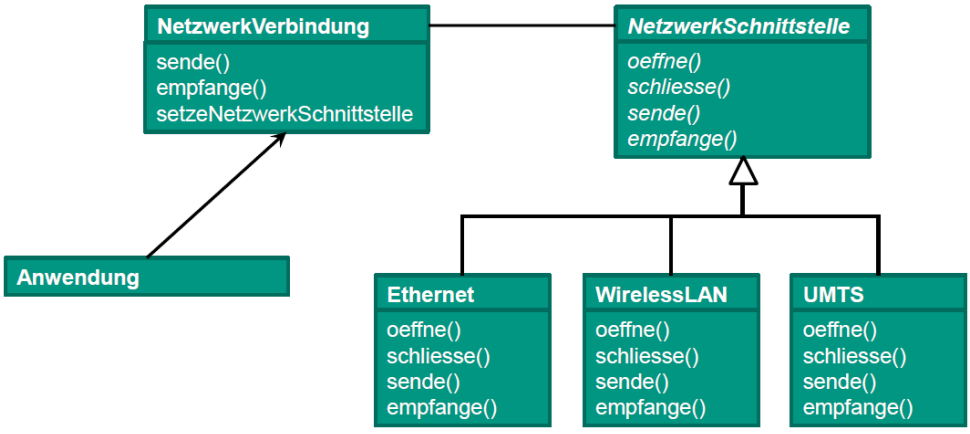
> Definiert eine Familie von Algorithmen, kapselt sie und macht sie austauschbar. Ermöglicht es, den Algorithmus unabhängig von nutzenden Klienten zu variieren.

- Manchmal müssen Algorithmen, abhänging von der notwendigen Performanz, der Menge oder des Typs der Daten, variiert werden.
- Anwendbar, wenn sich viele verwandte Klassen nur in ihrem Verhalten unterscheiden. Strategieobjekte bieten die Möglichkeit, eine Klasse mit einer von mehreren möglichen Verhaltensweisen zu konfigurieren.
- Um Fallunterscheidungen für Verhaltensweisen abzufangen und damit switch-less-prorgamming zu vollziehen.

Allgemeine Struktur



Beispiel



Java verwendet es zur Implementierung der Anordner (LayoutManager) in AWT und Swing.

5 Zustandshandhabungsmuster

- Die Muster dieser Kategorie bearbeiten den Zustand von Objekten, unabhängig von deren Zweck.

Die ZustandshandhabungsMuster:

- *Einzelstück:* Sichert zu, dass eine Klasse genau ein Exemplar besitzt und stellt einen globalen Zugriffspunkt darauf bereit.
- *Fliegengewicht:* Nutzt Objekte kleiner Granularität gemeinsam, um große Mengen von ihnen effizient speichern zu können.
- *Memento:* Erfasst und externalisiert den internen Zustand eines Objekts, ohne seine Kapselung zu verletzten, so dass das Objekt später in diesen Zustand zurückversetzt werden kann.
- *Prototyp:* Bestimmt die Arten zu erzeugender Objekte durch die Verwendung eines typischen Exemplars und erzeugt neue Objekte durch Kopieren diesen Prototyps.
- *Zustand:* Ändert das Verhalten das Objektes, wenn sich dessen interner Zustand ändert.

5.1 Einzelstück

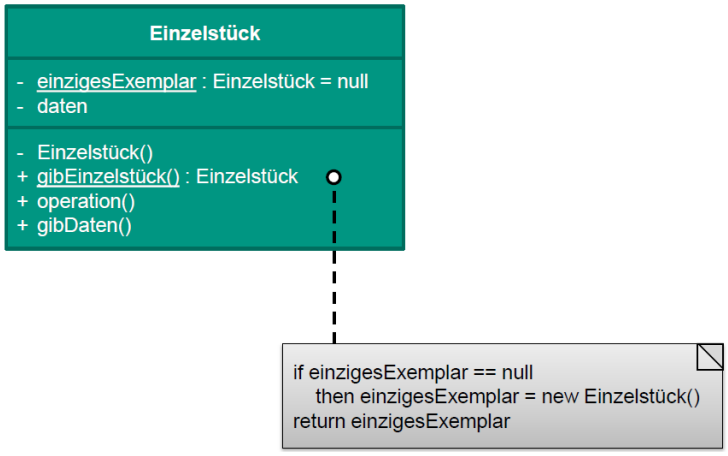
oder auch singleton.

Sichert zu, dass eine Klasse genau ein Exemplar besitzt und stellt einen globalen Zugriffspunkt darauf bereit.

Die Klasse ist selbst für die Verwaltung ihres einzigen Exemplars zuständig. Die klasse kann durch Abfangen von Befehlen zur Erzegung neuer Objekte sicherstellen, dass kein weiteres Exemplar erzeugt wird.

Anwendbar, wenn es eben nur eine einzige Instanz darf oder wenn die einzige Instanz durch Unterklassenbildung erweiterbar sein soll und die Klienten ohne Veränderung ihres Quelltextes diese nutzen sollen.

Allgemeine Struktur



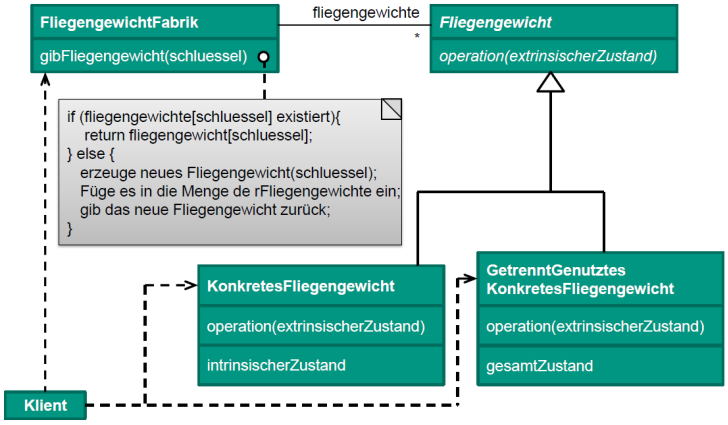
5.2 Fliegengewicht

oder auch flyweight.

> Nutzt Objekte kleiner Granularität gemeinsam, um große Mengen von ihnen effizient speichern zu können.

Das Fliegengewicht wird verwendet, wenn eine große Anzahl von Objekten benötigt wird, die sich bestimmte variable Informationen teilen und eine herkömmliche Implementierung unverhältnismäßig viele Ressourcen erfordern und alleine die Anzahl zu Problemen führen würde. Ein Teil des Zustandes dieser Objekte kann in den Kontext ausgelagert werden (extrinsisch). Nach der Entfernung des Zustandes reduziert sich die Anzahl verschiedener Objekte auf ein überschaubares Maß.

Allgemeine Struktur



Das Fliegengewicht ist abstrakt und definiert die Schnittstelle für Objekte, die einen von außen sichtbaren Zustand empfangen und verarbeiten. Das konkrete Fliegengewicht implementiert die Fliegengewichtschnittstelle. Bei Bedarf wird ein innerer Zustand ergänzt. Exemplare von KonkretesFliegengewicht oder abgeleiteten Klassen werden gemeinsam genutzt. Der intrinsische Zustand muss unabhängig vom Kontext sein.

Das getrennt genutzte konkrete Fliegengewicht implementiert diese Schnittstelle ebenfalls, enthält allerdings den kompletten Zustand. Das bedeutet, dass diese Objekte nicht gemeinsam genutzt werden. Hierbei handelt es sich nicht mehr im engeren Sinne um Fliegengewichte. Es können sich sogar echte „Schwergewichte“ dahinter verbergen. Es zeigt vielmehr die Stelle, an der „normale“ Objekte ihren Platz in dem Muster finden.

Die Fliegengewicht-Fabrik erzeugt und verwaltet Fliegengewichte. Sie stellt auch die korrekte Benutzung der gemeinsam benutzten Objekte sicher. Der Klient verwaltet Referenzen auf Fliegengewichte und den extrinsischen Zustand der Fliegengewichte.

Beispiel

Objektmodellierung in einem Texteditor bis hinunter zu einzelnen Zeichen. Die einzelnen Zeichen können durch einen Code repräsentiert werden (innerer, „intrinsicher“ Zustand). Die Informationen über Schriftart, Größe und Position können extranilisiert werden (äußerer, „extrinisicher“ Zustand) und in dem Zeilen- oder Spaltenobjekt oder auch in Teilfolgen von Zeichen gespeichert werden.

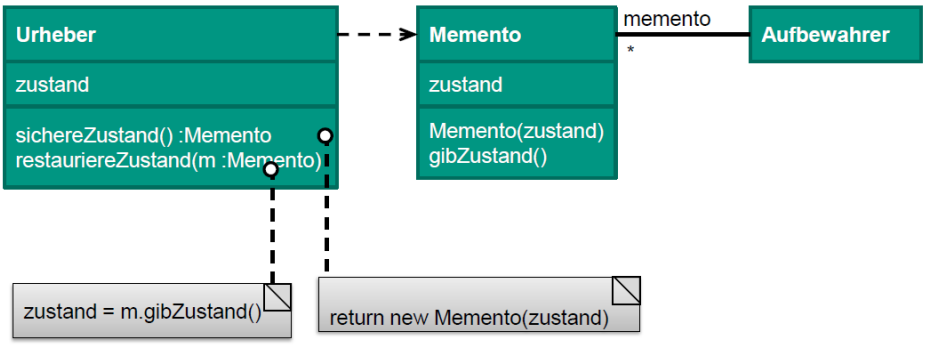
5.3 Memento

oder auch Token.

> Erfasst und externalisiert den internen Zustand eines Objekts, ohne seine Kapselung zu verletzten, so dass das Objekt später in diesen Zustand zurückversetzt werden kann.

Kann verwendet werden, wenn eine Momentaufnahme (eines Teils) des Zustands eines Objekts zwischengespeichert werden muss, so dass es zu einem späteren Zeitpunkt in diesen Zustand zurückversetzt werden kann, und wenn eine direkte Schnittstelle zum Ermitteln des Zustands die Implementierungsdetails offenlegen und die Kapselung des Objekts aufbrechen würde. Außerdem stellt das Muster eine einfache Möglichkeit dar, eine teilweise Schnittstelle zu schaffen.

Allgemeine Struktur



Beispiel

In Spielen zur Spielstandsspeicherung.

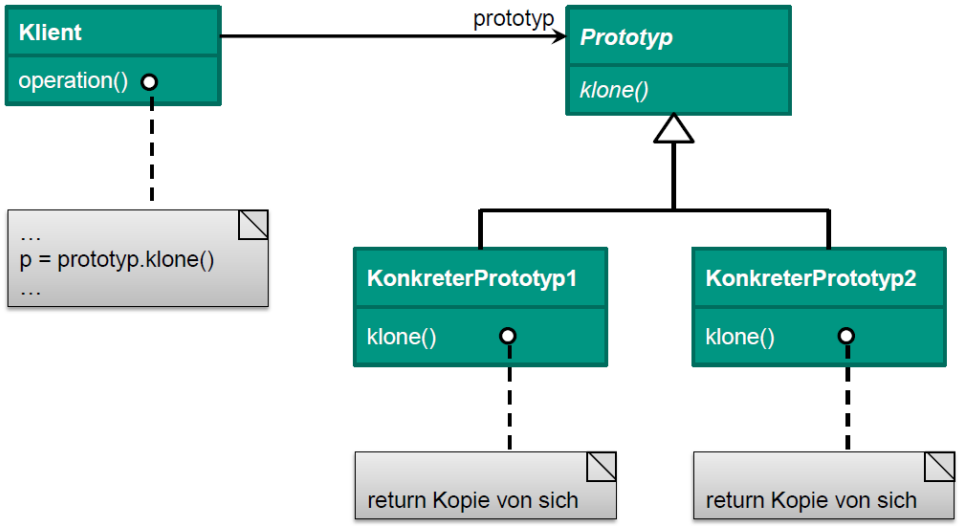
5.4 Prototyp

oder auch prototype.

> Bestimmt die Arten zu erzeugender Objekte durch die Verwendung eines typischen Exemplars und erzeugt neue Objekte durch Kopieren diesen Prototyps.

- Das Prototypenmuster wird verwendet, wenn ein System unabhängig davon sein soll, wie seine Produkte erzeugt, zusammengesetzt und repräsentiert werden.
- Falls der Aufbau eines Objekts wesentlich mehr Zeit erfodert als eine Kopie anzulegen.
- Wenn die Klassen zu erzeugender Objekte erst zur Laufzeit spezifiziert werden, z.B. durch dynamisches Laden.
- Um eine Klassenhierarchie von Fabriken zu vermeiden, die parallel zur Klassenhierarchie der Produkte verläuft.
- Wenn Exemplare einer Klasse nur wenige Zustandkombinatonen haben können. Bequemer eine entsprechende Anzahl von Prototypen zu klonen als von Hand jedesmal neu zu erzeugen.

Allgemeine Struktur



Beispiel

In der Dokumentenverarbeitung (z.B. Microsoft Office, LibreOffice) ist das Prototyp-Muster Standard: Die Vorlagen werden mit denselben Tools wie die eigentlichen Dokumente bearbeitet; zum Erstellen eines Dokuments wird eine Vorlage kopiert und dann weiterbearbeitet.

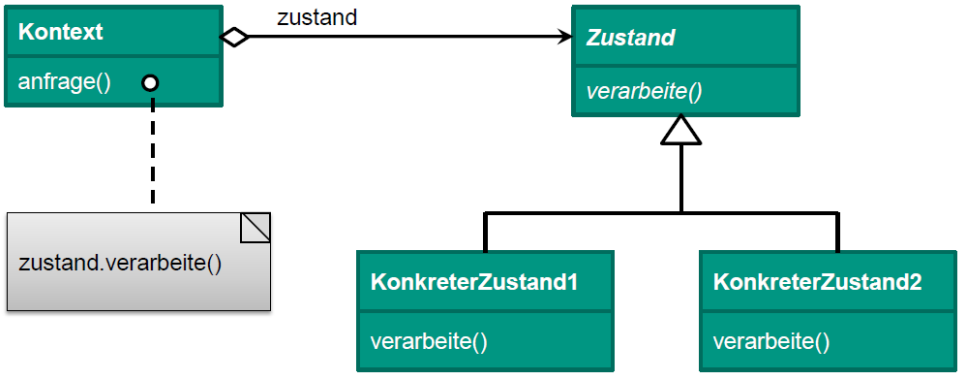
Java Objekt und clone().

5.5 Zustand

oder auch state.

Ändert das Verhalten das Objektes, wenn sich dessen interner Zustand ändert.

Allgemeine Struktur



6 Steuerungsmuster

- Steuern des Kontrollflusses.
- Bewirken, dass zur richtigen Zeit die richtigen Methoden aufgerufen werden.

Die Steuerungsmuster:

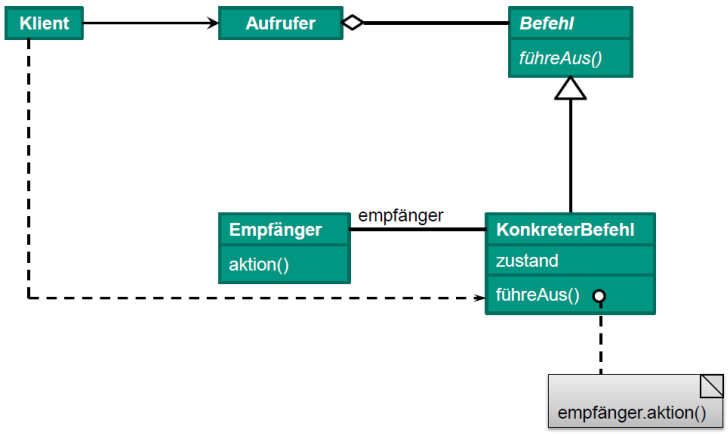
- *Befehl:* Kapselt einen Befehl als ein Objekt. Dies ermöglicht es, Klienten mit verschiedenen Anfragen zu parametrisieren, Operationen in eine Warteschalnge zu stellen, ein Logbuch zu führen und Operationen rückgängig zu machen.
- *Master-Worker:* Ein Auftraggeber verteilt die Arbeit an identische Arbeiter und berechnet das Endergbenis aus den Teilergebnissen, welche die Arbeiter zurückliefern. Bietet fehlertolerante und parallele Berechnung.

6.1 Befehl

oder auch command, Kommando, Aktion, Transaktion.

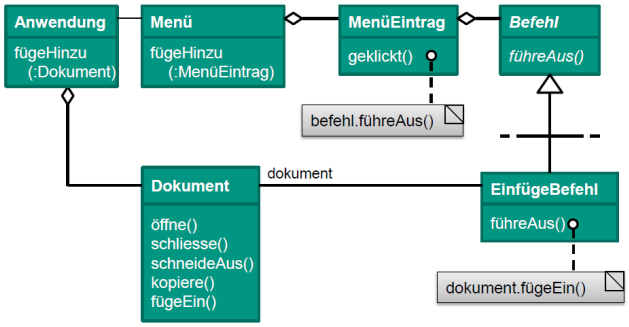
> Kapselt einen Befehl als ein Objekt. Dies ermöglicht es, Klienten mit verschiedenen Anfragen zu parametrisieren, Operationen in eine Warteschalnge zu stellen, ein Logbuch zu führen und Operationen rückgängig zu machen.

Allgemeine Struktur



Beispiel

Ein Befehl enthält eine Methode `führeAus()` und speichert das Objekt, an dem diese Operation durchgeführt werden soll.



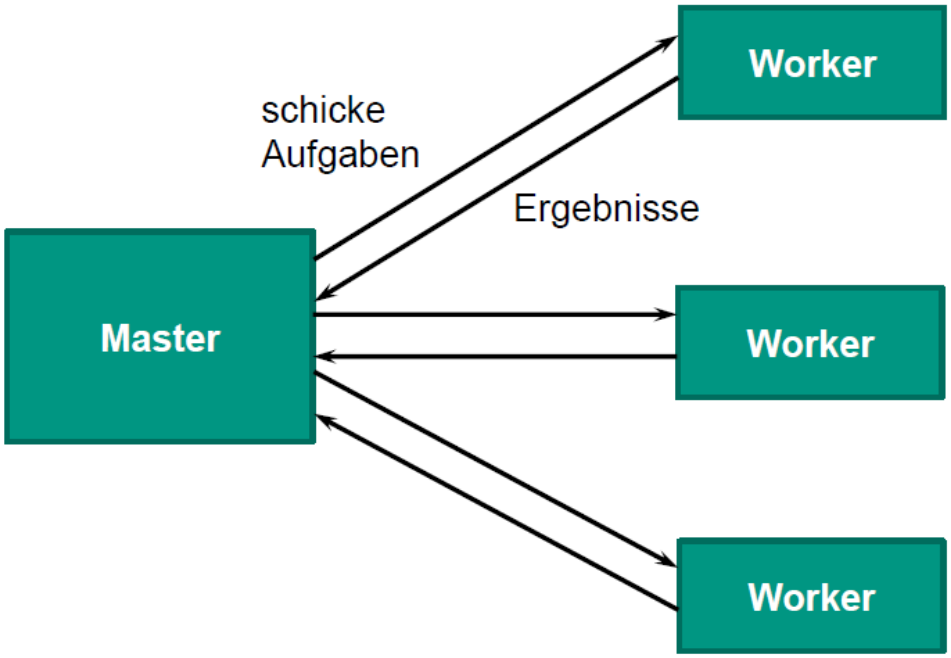
6.2 Master-Worker

oder auch Auftraggeber/-nehmer, Master/Slave.

> Ein Auftraggeber verteilt die Arbeit an identische Arbeiter und berechnet das Endergbenis aus den Teilergebnissen, welche die Arbeiter zurückliefern. Bietet fehlertolerante und parallele Berechnung.

Zur Parallelisierung mehrere unabhängiger Aufgaben durch Verteilung auf mehrere verfügbare Prozesse. Kann zum Ausgeglichen der Belastung der Arbeiter genutzt werden.

Allgemeine Struktur



7 Bequemlichkeitsmuster

- ✠ Diese Muster sparen etwas Schreib- oder Denkarbeit.

Die Bequemlichkeitsmuster:

- ❋ *Bequemlichkeitsklasse:* Vereinfacht den Methodenaufruf durch Bereithaltung der Parametr in einer speziellen Klasse.
- ❋ *Bequemlichkeitsmethode:* Vereinfacht den Methodenaufruf durch die Bereitstellung häufig genutzer Parameterkombinationen in zusätzlichen Methoden (Überladen).
- ❋ *Fassade:* Bietet eine einheitliche Schnittstelle zu einer Menge von Schnittstellen eines Subsystems. Die Fassadenklasse definiert eine abstrakte Schnittstelle, welche die Benutzung des Subsystems vereinfacht.
- ❋ *Null-Objekt:* Stellt einen Stellvertreter zur Verfügung, der die gleiche Schnittstele bietet, aber nichts tut. Das Null-Objekt kapselt die Implementierungs-Entscheidung (wie genau es „nichts tut“) und versteckt diese Details vor seinen Mitarbeitern.

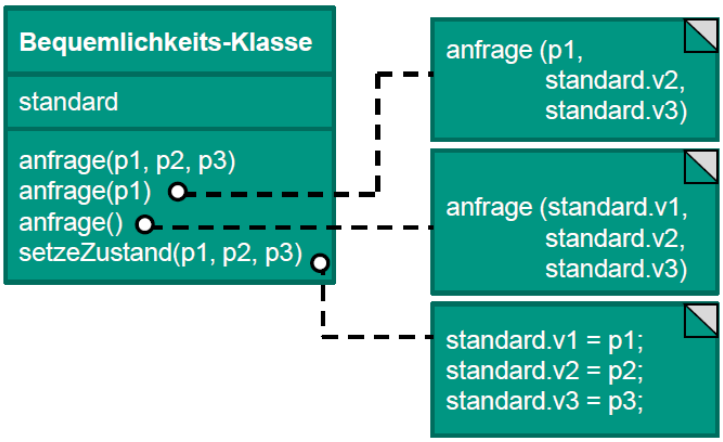
7.1 Bequemlichkeits-Klasse

oder auch convenience class.

Vereinfacht den Methodenaufruf durch Bereithaltung der Parametr in einer speziellen Klasse.

Anwendbar, wenn Methoden häufig mit den gleichen Parametern aufgerufen werden, die sich nur selten ändern.

Allgemeine Struktur



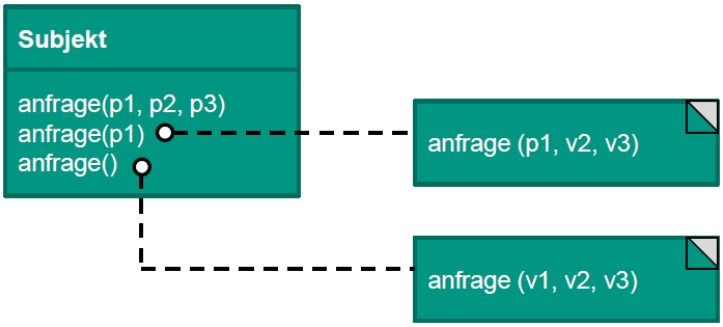
7.2 Bequemlichkeits-Methode

oder auch convenience method, vorbelegte Parameter, default parameters.

Vereinfacht den Methodenaufruf durch die Bereitstellung häufig genutzer Parameter-kombinationen in zusätzlichen Methoden (Überladen).

Anwendbar, wenn Methodenaufrufe häufig mit den gleichen Paramtern auftretten.

Allgemeine Struktur



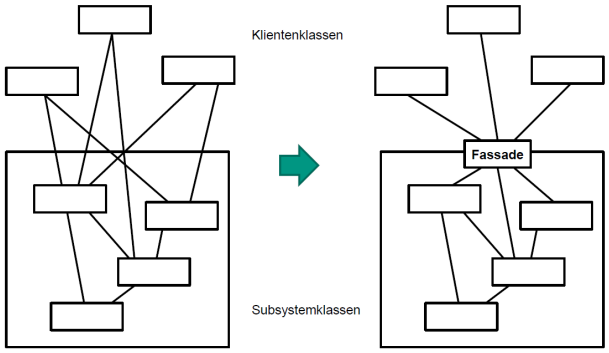
7.3 Fassade

oder auch facade.

> Bietet eine einheitliche Schnittstelle zu einer Menge von Schnittstellen eines Subsystems. Die Fassadenklasse definiert eine abstrakte Schnittstelle, welche die Benutzung des Subsystems vereinfacht.

Sollte eingesetzt werden, wenn eine einfache Schnittstelle zu einem komplexen Subsystem angeboten werden soll. Fassade reicht dem Klienten als Sicht auf das Subsystem. Zur Entkoppelung von Klienten und Subsystemen. Wenn Subsystem in Schichten aufgeteilt werden soll, verwendet man Fassaden, um einen Eintrittspunkt zu jeder Subsystemschicht zu definieren.

Beispiel



7.4 Null-Objekt

> Stellt einen Stellvertreter zur Verfügung, der die gleiche Schnittstele bietet, aber nichts tut. Das Null-Objekt kapselt die Implementierungs-Entscheidung (wie genau es „nichts tut“) und versteckt diese Details vor seinen Mitarbeitern.

So kann verhindert werden, dass der Code mit Tests gegen Null-Werte verschmutzt wird.

Verwendung kann es finden, wenn ein Objekt Mitarbeiter benötigt und einer oder mehrere von ihnen nichts tut. Wenn Klienten sich nicht um den Unterschied zwischen einem echten Mitarbeiter und einem der nichts tut kümmern sollen. Zum Beispiel auch für Adapterklassen.

Allgemeine Struktur