MAYFLOWER

Decorator Pattern

Pattern DoJo, 16. März 2021

Definition

Das Decorator Pattern ermöglicht die Ausstattung von Objekten mit zusätzlichen Funktionalitäten indem es in spezielle Wrapper-Objekte gefasst wird, welche dieses Verhalten beinhalten.

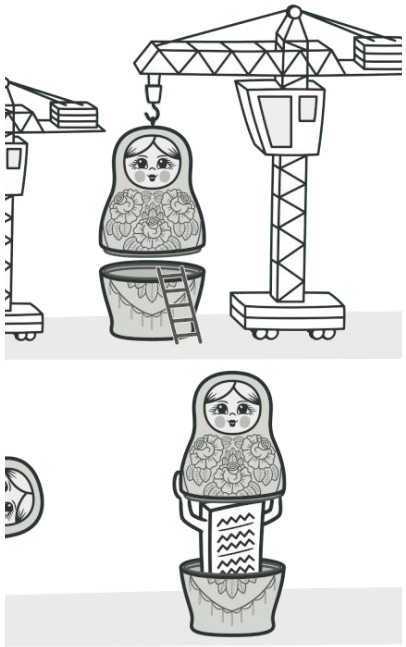
Das Decorator Pattern gehört zur Gruppe der Strukturmuster da es eine bestimmte Strukturierung unserer Klassen und Schnittstellen vorgibt.

Imagine ist begeistert von den Möglichkeiten objekt-orientierter Sprachen!

Imagine und ihr Team sollen ein Bestellsystem für eine große Pizzarestaurantkette entwickeln.

Neben verschiedenen Pizzasorten sollen beliebige Beläge unabhängig zur Verfügung stehen und beliebig kombiniert werden.

Alle handelnden Personen sind frei erfunden



Problem

Die Entität Pizza soll mit der Entität Topping unabhängig voneinander aber beliebig oft miteinander kombiniert werden.

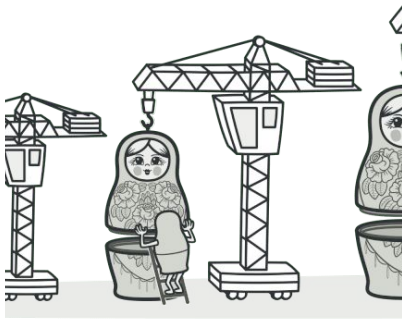
Mit dem Decorator Pattern kann Imagine die objektorientierte Konzeptionierung der Sprache in ihrer Anwendung voll entfalten.

Decorator1.kt

Neue Anforderungen

Nun sollen sowohl neue Pizzasorten, neue Pizzabeläge sowie ein Abrechnungssystem für alle Objekte eingeführt werden.

Mit dem verbauten Decorator Pattern kann Imagine diese Erweiterungen schnell realisieren.



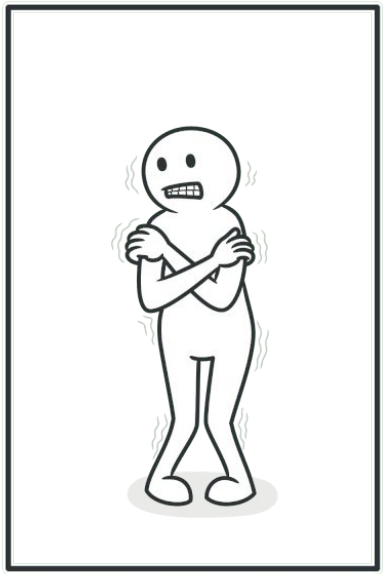
Lösung

Weitere Subklassen von Pizza und Topping können jederzeit und unabhängig voneinander eingeführt, verwendet und kombiniert werden.

Decorator2.kt

Technische Schulden?

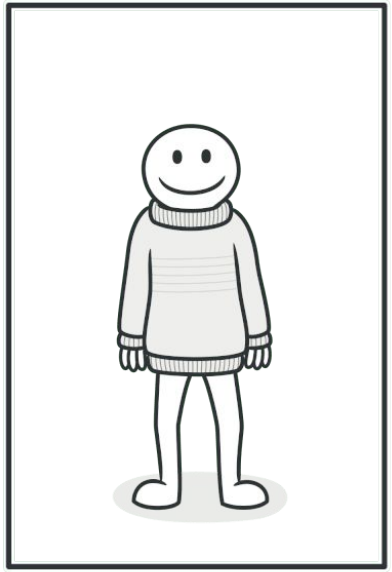
Der Code arbeitet korrekt und Imagine hat das Pattern richtig angewandt. Aber ist der Einsatz des Decorator Patterns wirklich überlegt? Hierüber gibt es unterschiedliche Ansichten.



Anwendbarkeit

- Das Pattern kann angewendet werden wenn es schwierig oder unmöglich ist, das Verhalten eines Objekts mittels Vererbung zu erweitern.

- Single Responsibility Prinzip: Eine monolithische Klasse mit vielen unterschiedlichen Verhaltensweisen kann in kleinere separate Einheiten zerlegt werden.



Verwendung

- Mit dem Pattern kann die Business Logik in Ebenen strukturiert werden, ein Decorator für jede Ebene erstellt und Objekte mit beliebigen Kombinationen dieser Business-Logiken erstellt werden.

- Es können mehrere unterschiedliche Verhaltensweisen kombiniert werden indem ein Objekt in mehrere Dekoratoren gewrappt wird.

Implementierung

1. Die Instanz eines Dekorierers wird vor die zu dekorierende Klasse geschaltet.

2. Der Dekorierer implementiert die gleiche Schnittstelle wie die zu dekorierende Klasse.

3. Aufrufe an den Dekorierer werden dann verändert oder unverändert weitergeleitet (Delegation), oder sie werden komplett in Eigenregie verarbeitet.

Imagine hat Spaß am Decorator Pattern!

Sie hat mit dem Einsatz des Decorator Patterns eine flexible Struktur geschaffen die für den Anwendungsfall gut passt und flexibel kombiniert und restrukturiert werden kann.

Damit kann die Anwendung auch in Zukunft gesund wachsen.

Vorteile

- Beliebig viele Dekorierer können nacheinander geschaltet werden.
- Die Dekorierer können zur Laufzeit und auch nach der Instanziierung ausgetauscht werden.
- Das Pattern stellt eine flexiblere Alternative zur Bildung von Unterklassen dar.

Nachteile

- Das Pattern fördert Close Coupling und die mehrfache Vererbung von Klassen.
- Bei der Verwendung von dekorierten Komponenten müssen die Nachrichten vom Dekorierer an das dekorierte Objekt weitergeleitet werden.
- Es ist schwierig, einen Dekorierer hinzuzufügen dessen Verhalten unabhängig von der aufgerufenen Reihenfolge im Dekorierer Stack sein soll.

Ja zum Decorator Pattern!

Das Decorator Pattern findet häufig Einsatz in API Code – Beispiele wären die Java Stream API oder Java AWT API.

Developer, Würzburg

Isoliert verbaut kann das Pattern in eigenen APIs gewinnbringend eingesetzt werden.

Developer, München

Zitate dienen nur Veranschaulichungszwecken

Nein zum Decorator Pattern!

Das Pattern tut aber ist schwer lesbar. Die Verwendung wurde im Team Alignment abgelehnt.

Developer, Berlin

Es kann schwer sein, die Struktur der Decorator im Nachhinein zu verändern.

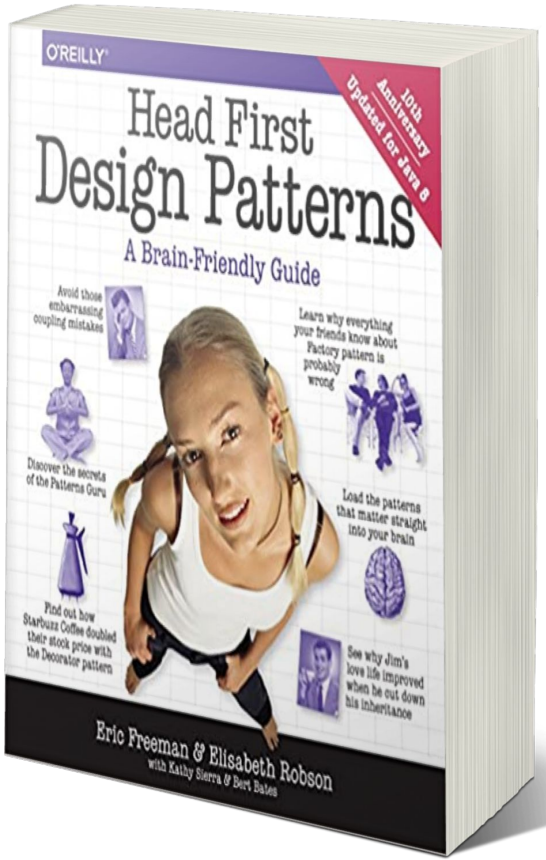
Developer, Würzburg

Zitate dienen nur Veranschaulichungszwecken

Vielen Dank für Ihre Aufmerksamkeit!

Die wichtigsten Design Patterns werden in dem modernen Klassiker Head First Design Patterns ausführlich behandelt.

Mit dem einzigartigen Head First Lernkonzept gelingt es den Autoren, die Materie witzig, leicht verständlich und dennoch gründlich darzustellen. Das ist nicht nur unterhaltsam, sondern auch effektiv.



Quellen und mehr Info

O’Reilly - Head First Design Patterns
Decorator Pattern - Refactoring Guru
Wikipedia - Decorator
Google Slides Template “Zündende Idee”
Try the Kotlin Programming Language!
Code Beispiele auf GitHub


Du kannst mithelfen, das Wissen über Modernisierungs- und Design-Patterns weiter zu verbessern, indem Du Dein Wissen in der Gruppe Dev teilst.