**VISITOR PATTERN**

Das Visitor Design Pattern ist eines der Verhaltensmuster. Es wird verwendet, wenn wir eine Operation an einer Gruppe ähnlicher Objekte ausführen müssen. Mit dem Visitor-Muster können wir die logischen Operationen von den Objekten in eine andere Klasse verschieben.

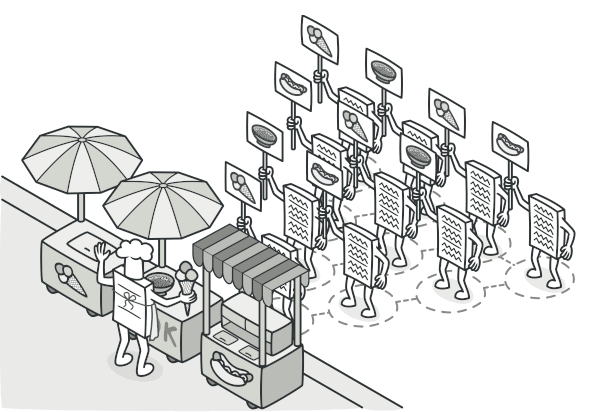


Bild: Visitor ist ein Verhaltensentwurfsmuster, mit dem Sie Algorithmen von den Objekten trennen können, mit denen sie arbeiten.

Das Visitor-Muster besteht aus zwei Teilen:

* Eine Methode namens Visit(), die vom Visitor implementiert wird und für jedes Element in der Datenstruktur aufgerufen wird.
* Besuchbare Klassen, die Accept()-Methoden bereitstellen, um einen Visitor zu akzeptieren.

…………….UML Diagram.

Mit dem Visitor Design Pattern können Sie einen Algorithmus von der Objektstruktur trennen, auf die er angewendet wird. Dieses Muster(pattern) ist nützlich, um vorhandenen Objektstrukturen neue Operationen hinzuzufügen, ohne sie zu ändern.

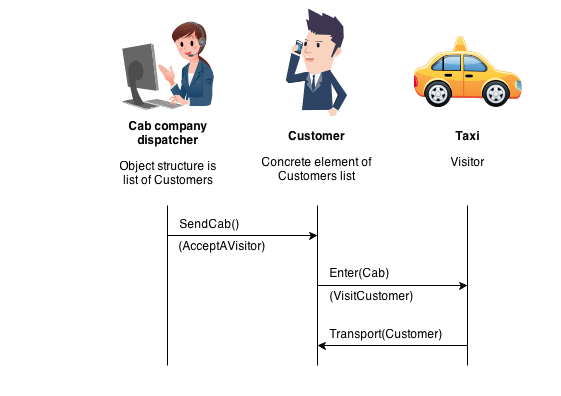
Visitor-Muster in der realen Welt

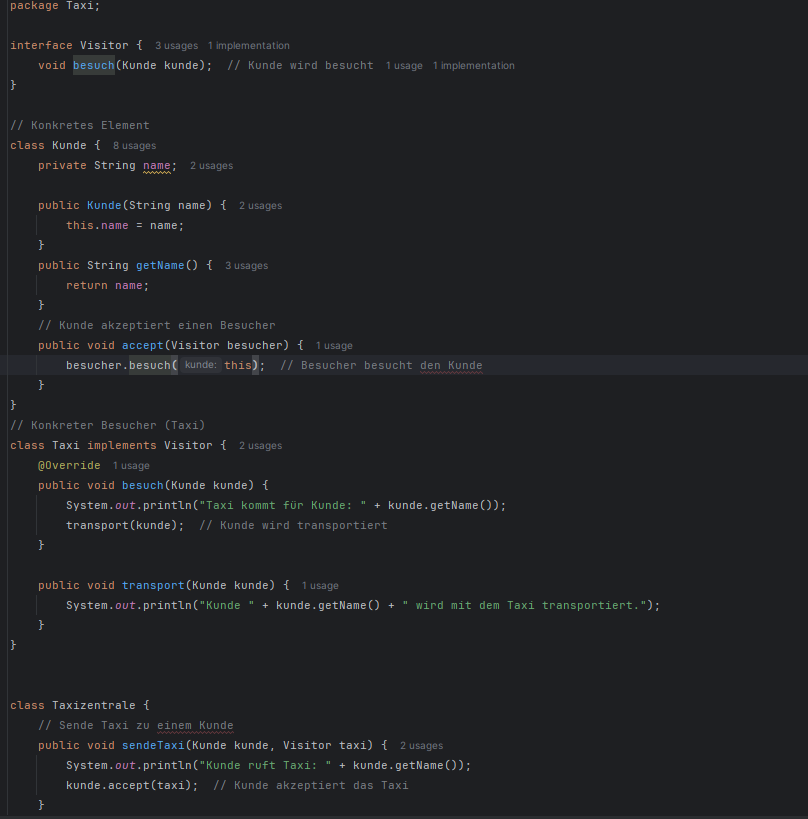
Eine reale Analogie hilft oft beim Verständnis eines Entwurfsmusters. Ein Beispiel für das Visitor-Muster ist das Taxi-Beispiel. Der Kunde bestellt ein Taxi, das vor der Tür ankommt. Wenn die Person einsteigt, hat das Taxi die Kontrolle über den Transport dieser Person.

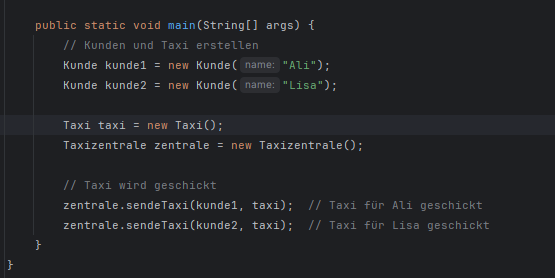
Einkaufen im Supermarkt ist ein weiteres gängiges Beispiel. Der Einkaufswagen ist Ihre Sammlung von Elementen. Wenn Sie an die Kasse kommen, handelt der Kassierer als Besucher. Er nimmt die verschiedenen Artikel (Ihre Einkäufe), einige mit Preisen und andere, die gewogen werden müssen, um Ihnen einen Gesamtbetrag zu nennen.

( Als Aufgabe können Sie gerne implementieren) 😉

Es ist schwierig, dieses Pattern in der realen Welt zu erklären, aber die Dinge werden klarer, wenn wir die Musterdarstellung durchgehen und sehen, wie wir es im Code verwenden.







Wo sollte ich dieses Pattern verwenden?

* Das Muster sollte verwendet werden, wenn Sie verschiedene und unabhängige Operationen auf einer Struktur von Objekten durchführen möchten. Dadurch vermeiden Sie es, Code in die Objektstruktur einzufügen, der besser getrennt bleiben sollte. Es fördert saubereren Code. Möglicherweise möchten Sie Operationen gegen eine Menge von Objekten mit unterschiedlichen Schnittstellen ausführen. Besucher sind auch nützlich, wenn Sie mehrere unabhängige Operationen über die Klassen hinweg durchführen müssen.
* Zusammenfassend lässt sich sagen, dass das Visitor-Muster wahrscheinlich die beste Wahl ist, wenn Sie einige logische Codes von den Elementen trennen möchten, die Sie als Eingabe verwenden.

Vorteile:

* Neue Operationen lassen sich leicht durch die Definition neuer Visitor(Besucher) hinzufügen
* Verwandte Operationen werden im Visitor(Besucher) zentral verwaltet und von besucherfremden Operationen getrennt
* Besucher können mit Objekten aus voneinander unabhängig Klassenhierarchien arbeiten

Nachteile:

* Die gute Erweiterungsmöglichkeit der Klassen von Besuchern muss mit einer schlechten Erweiterbarkeit der Klassen der konkreten Elemente erkauft werden.

Müssen neue konkrete Elemente hinzugefügt werden, so führt dies dazu, dass viele Visitor-visitor-Methoden implementiert werden müssen.

**Kodkomplexität**: Mit der Zunahme der Besucherklassen steigt die Komplexität des Codes, was die Wartungskosten erhöhen kann

😊 Fragen zu stellen ist nicht ERFORDERLICH 😊