Entwurfsmuster Fassade

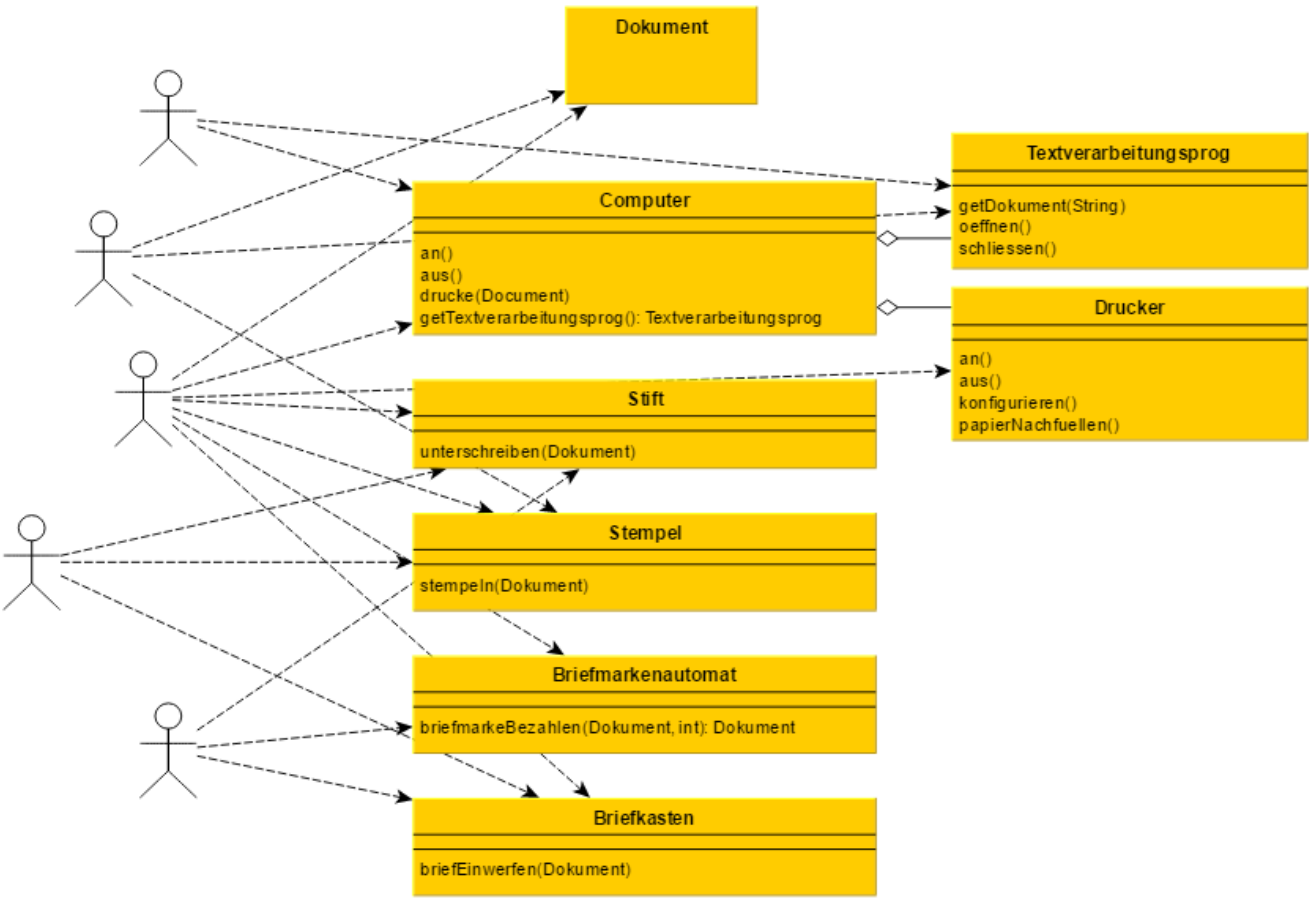
Fassade ist ein Entwurfsmuster zu der Kategorie der **Strukturmuster**. Es bietet eine einheitliche und meist **vereinfachte Schnittstelle zu einer Menge von Schnittstellen** eines Subsystems. Wenn ein Subsystem viele technisch orientierte Klassen enthält, die selten von außen verwendet werden, hilft es, eine Fassade zu verwenden. Die Fassade ist eine Klasse mit ausgewählten Methoden, die eine häufig benötigte Untermenge an Funktionalität des Subsystems umfasst. Die Fassadenklasse definiert **eine abstrakte Schnittstelle**, welche die Benutzung des Subsystems vereinfacht.

Vorteile und Nachteile

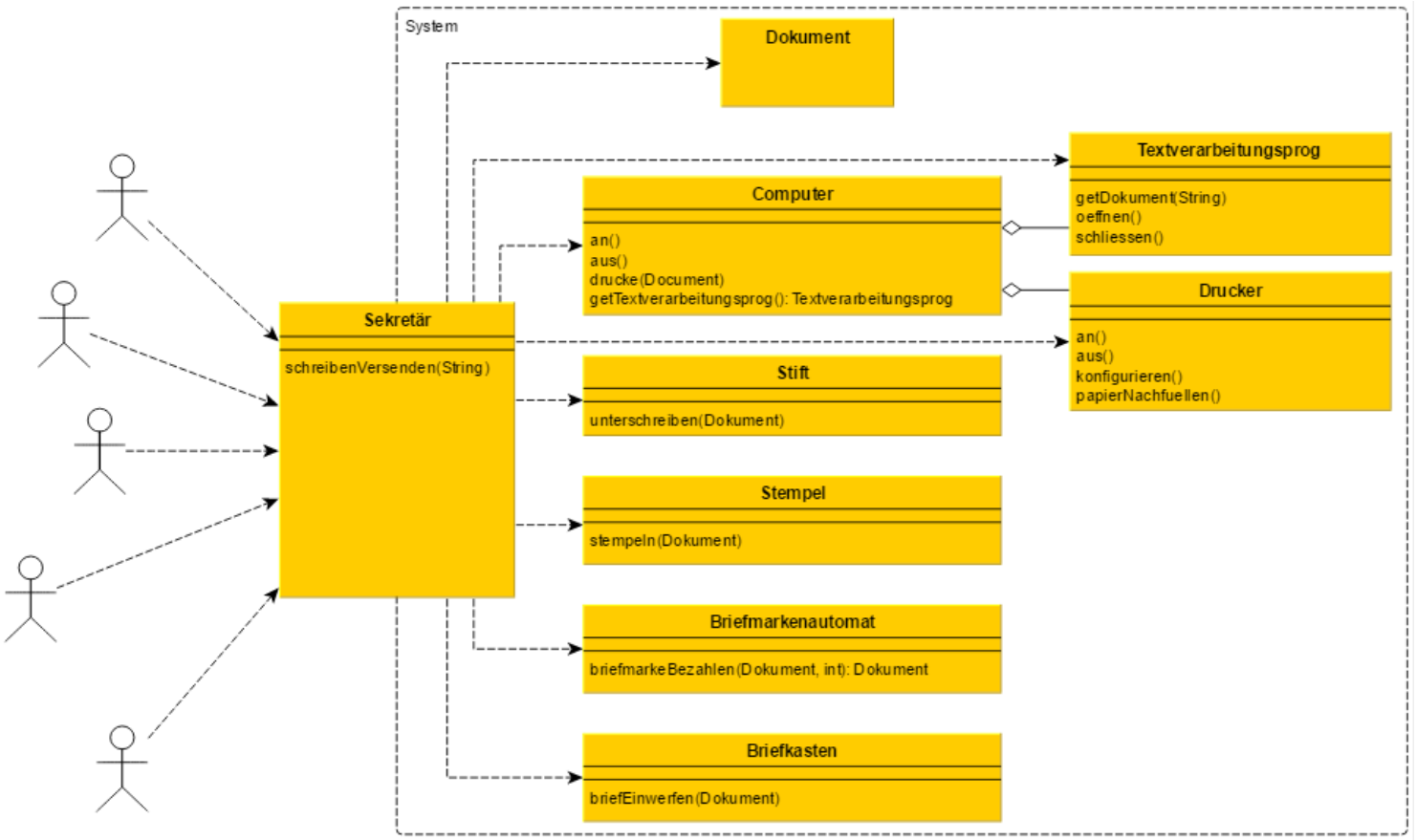
Die Fassade fördert die **lose Kopplung**, weil sie das zugrunde liegende Subsystem versteckt, und **senkt die Komplexität**, da mehrere Schnittstellen zu einer zusammengefasst werden. Außerdem kann das Subsystem durch die lose Kopplung **leichter erweitert** werden.

Der Nachteil besteht darin, dass zusätzlicher Ballast bei einfachen Subsystemen entstehen kann, oder auch die Funktionalität / Flexibilität eingeschränkt werden.

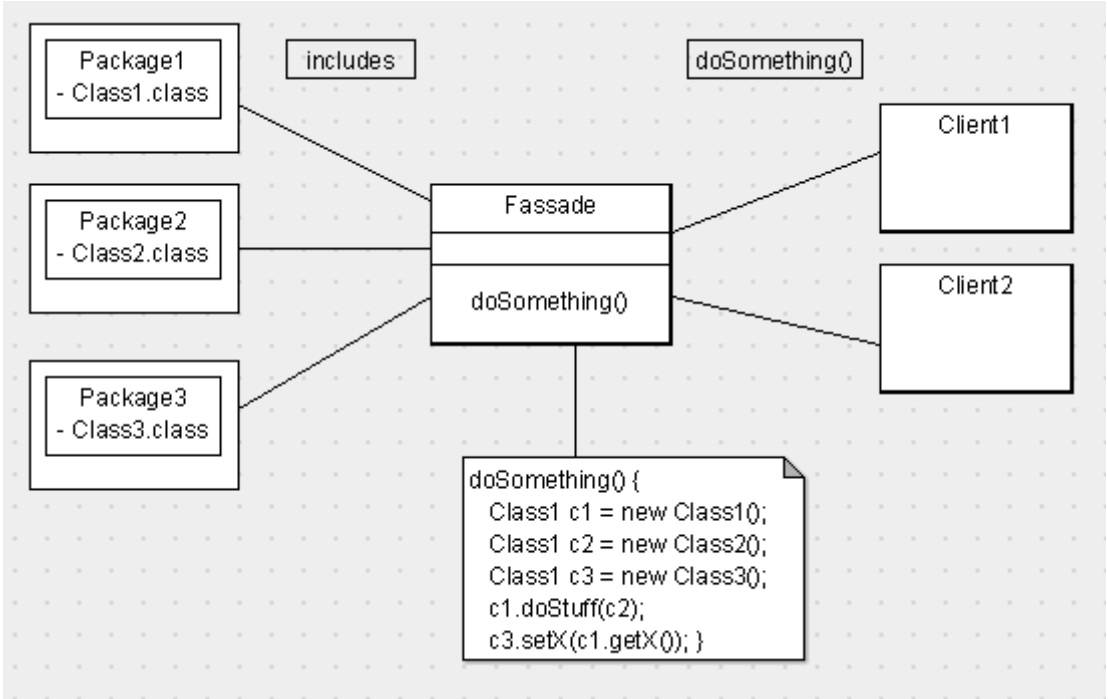
UML-Beispiel: Firmenverwaltung
Wenn es viele Abhängigkeiten zwischen den Klienten und den Implementierungsklassen einer Abstraktion gibt.



UML-Beispiel mit Fassade-Entwurfsmuster:
Die Einführung einer Fassade entkoppelt die Subsysteme von Klienten und anderen Subsystemen.



UML-Beispiel:
Wenn Subsysteme in Schichten aufgeteilt werden sollen, verwendet man eine Fassade, um einen Eintrittspunkt zu jeder Subsystemschicht zu definieren.



SOLID-Prinzipien

Das **Single-Responsibility-Prinzip** besagt, dass jede Klasse nur eine einzige Verantwortung haben solle.

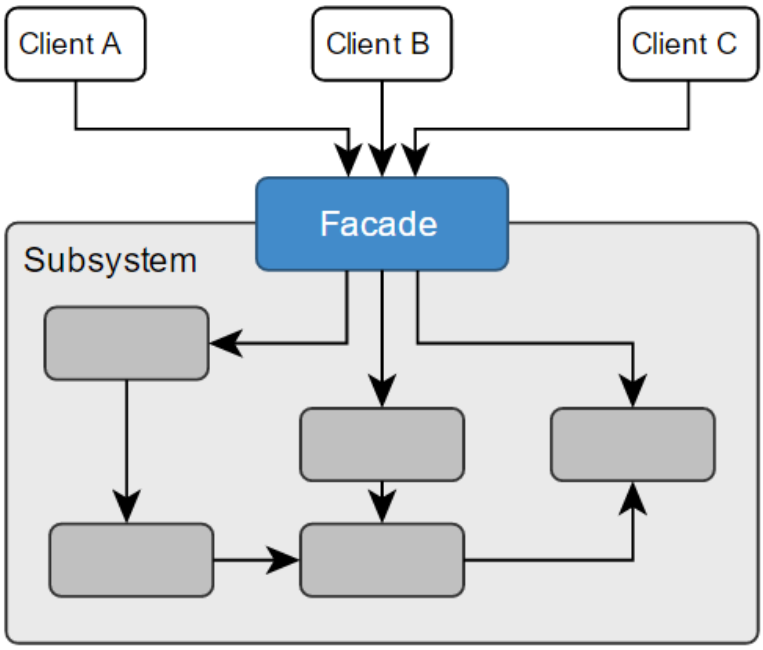
Das **Open-Closed-Prinzip** besagt, dass Software-Einheiten Erweiterungen möglich machen sollen, aber ohne dabei ihr Verhalten zu ändern. Eine Erweiterung wäre zum Beispiel die Vererbung.

Das **Liskovsche Substitutionsprinzip** (Ersetzungsprinzip) sagt aus, dass eine Unterklasse stets alle Eigenschaften der Oberklasse erfüllen und immer als Objekt der Oberklasse verwendbar sein muss. Eine Unterklasse darf Erweiterungen enthalten, nicht aber grundlegende Änderungen.

Das **Interface Segregation Prinzip** sagt aus, dass Software-Elemente („Clients“) sollten nicht von Schnittstellen abhängen, die sie nicht benötigen.

Das **Dependency Inversion Principle** besagt, dass in der Basis eines Entwurfs nicht die speziellen Eigenschaften von Modulen sind, sondern deren gemeinsame Merkmale sollen in einer gemeinsamen Schnittstelle abstrahiert werden. Durch die Abstraktion werden die nicht mehr relevanten Details eines Moduls ausgeblendet.

Eigenes Beispiel:



Anwendungsfälle

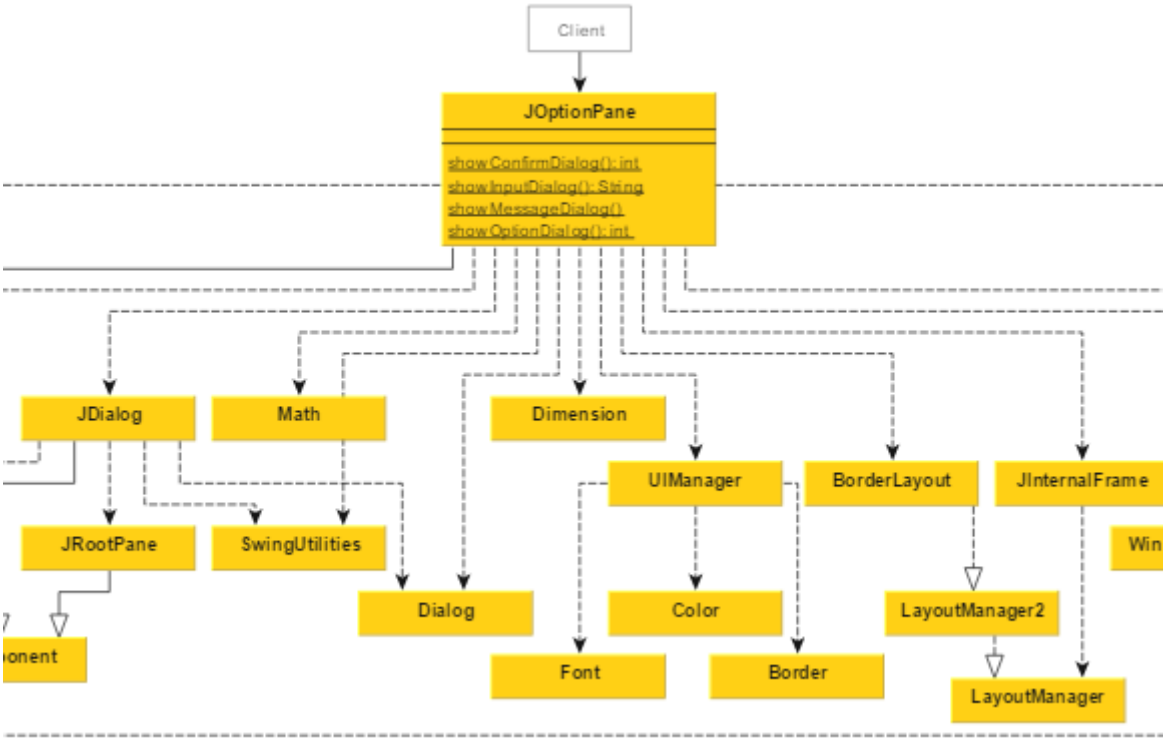
Das Facade Design Pattern kann angewandt werden, wenn:

1. eine vereinfachte Schnittstelle zur Nutzung eines komplizierten Subsystems oder Menge von Objekten benötigt wird.
2. die Reduzierung der Abhängigkeiten zwischen dem Client und dem benutzten Subsystem angestrebt wird.
3. ein System in Schichten unterteilt werden soll. Die Schichten kommunizieren nur noch über Fassaden, die den Zugang zum Subsystem darstellen. Somit wird das System unterteilt und die Abhängigkeiten zwischen den Teilsystemen gesenkt.

Anwendung in der Java Standardbibliothek

javax.swing.JOptionPane

Die Swing-Convenience-Klasse JOptionPane ist eine Fassade für Dialogfenster. Mit ihren statischen Methoden (showMessageDialog(), showInputDialog() etc.) stellt sie dem Client eine einfache Schnittstelle zur Erstellung von Dialogfenstern zur Verfügung. Der Client muss dabei keine Kenntnis von den verwendeten Swingkomponenten haben.



Quellen:

https://de.wikipedia.org/wiki/Fassade_(Entwurfsmuster)
https://www.philipphauer.de/study/se/design-pattern/facade.php
http://krsteski.de/php-tricks-und-tipps/das-facade-pattern.html
https://www.tutorialspoint.com/design_pattern/facade_pattern.htm