# ENTWURFSMUSTER 1

Arbeite dich in das Thema Entwurfsmuster ein. Du sollst wissen, was Entwurfsmuster sind, wie sie beschrieben  
werden und wozu man sie benötigt.  
Du bekommst dazu auch eine Einführung im Unterricht anhand des Strategy-Entwurfsmusters.

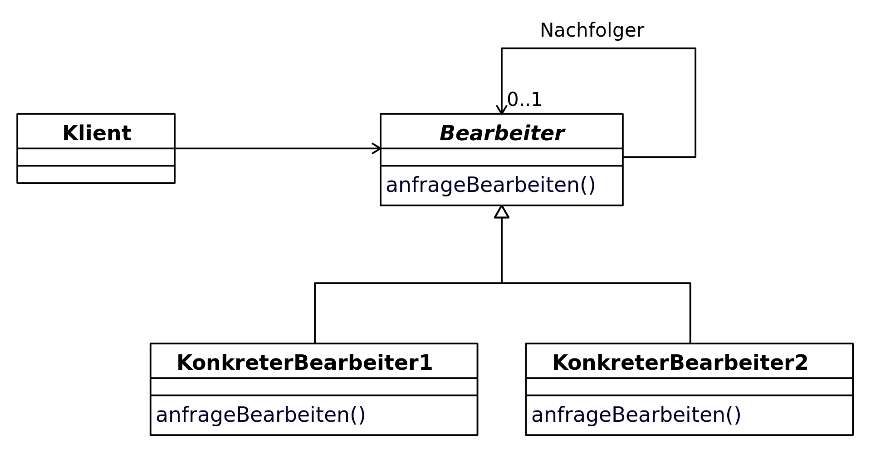
SOLID Prinzip:

* The single-responsibility principle: "There should never be more than one reason for a class to change." In other words, every class should have only one responsibility.
* The open–closed principle: "Software entities ... should be open for extension, but closed for modification."
* The Liskov substitution principle: "Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it." See also design by contract.
* The interface segregation principle: "Clients should not be forced to depend upon interfaces that they do not use."
* The dependency inversion principle: "Depend upon abstractions, [not] concretions."

Entwurfsmuster:

In objektorientierten Programmiersprachen werden Design Patterns verwendet, um ein ganz bestimmtes Software-Entwurfsproblem zu lösen. Das Design Pattern (Entwurfsmuster) dient dabei als eine Art Rezept, mit dessen Hilfe die gegebene Programmieraufgabe gelöst wird.

ENTWURFSMUSTER 2  
Arbeite dich in folgende Entwurfsmuster ein:  
• Chain of Responsibility

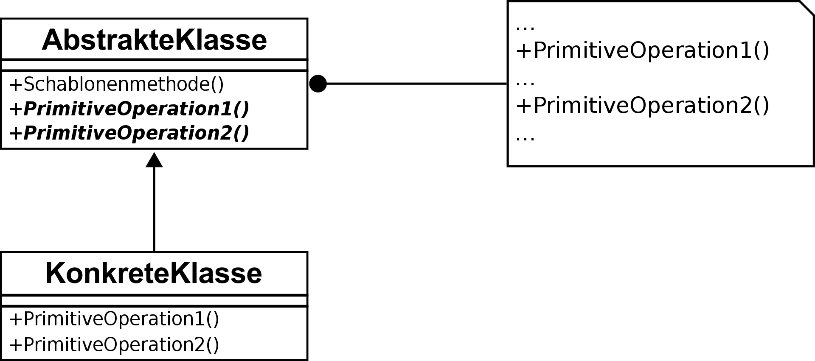


Bei der Chain of Responsibility werden mehrere Objekte hintereinander geschaltet (miteinander verkettet), um gemeinsam eine eingehende Anfrage bearbeiten zu können. Diese Anfrage wird an der Kette entlang geleitet, bis eines der Objekte die Anfrage beantworten kann. Der Klient, von dem die Anfrage ausgeht, hat dabei keine Kenntnis darüber, von welchem Objekt die Anfrage beantwortet werden wird.

Als Beispiel wird eine Kette mit den Grundrechnungsarten programmiert. Siehe dazu Github Repository. Dokumentation ist im Code enthalten.

<https://github.com/Manuel2103/FSE/tree/main/Mikroarchitektur>

• TemplateMethod



Beim Schablonenmethoden-Entwurfsmuster wird in einer abstrakten Klasse das Skelett eines Algorithmus definiert. Die konkrete Ausformung der einzelnen Schritte wird an Unterklassen delegiert. Dadurch besteht die Möglichkeit, einzelne Schritte des Algorithmus zu verändern oder zu überschreiben, ohne dass die zu Grunde liegende Struktur des Algorithmus modifiziert werden muss. Die Schablonenmethode (engl. template method) ruft abstrakte Methoden auf, die erst in den Unterklassen definiert werden. Diese Methoden werden auch als Einschubmethoden bezeichnet.

Zusätzlich können in der Schablonenmethode an bestimmten Stellen Hook-Operationen aufgerufen werden, deren Standardimplementierung in der abstrakten Klasse nichts tut. Auf diese Weise kann man an vordefinierten Stellen im Algorithmus zusätzliche Funktionalität einfügen.

• Decorator  
• Builder  
• Adapter  
• Observer