CS1 Task 13: Design pattern "Decorator"

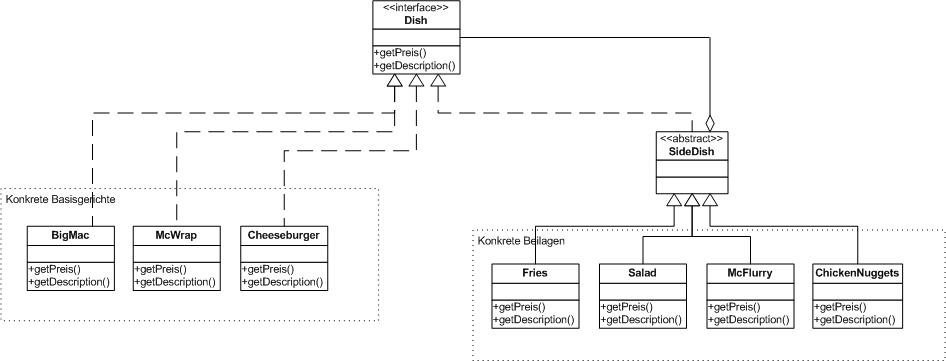
# Konzept

Das Decorator-Pattern ermöglicht das dynamische Hinzufügen von Fähigkeiten zu einer Klasse. Dazu wird die Klasse, dessen Verhalten wir erweitern möchten (Component), mit anderen Klassen (Decorator) "dekoriert". Der Decorator ist vom selben Klassentyp wie die Component und besitzt somit dieselben Schnittstellen.

Eine Component kann mit beliebig vielen Decorators dekoriert werden, um so seine Fähigkeiten immer weiter auszubauen.

Als Beispiel dient McDonalds, bei dem man verschiedene Basisgerichte mit vielen verfügbaren Beilagen kombinieren kann. Ohne Pattern würde man wohl für jede Kombination eine eigene Klasse erstellt werden, damit Preis und Beschreibung korrekt ausgegeben werden könnte.

Mit dem Decorator-Pattern können die Basisgerichte (Component) mit beliebig vielen Beilagen (Decorator) erweitert werden.



|  |  |
| --- | --- |
|  |  |
|  |  |
|  | |

# Vor-/Nachteile

## Vorteile

* Dynamik und Flexibilität:  
  Klassen können während der Laufzeit ergänzt um Funktionalitäten werden. Ausserdem sind auch Mehrfachkombinationen möglich.
* Erweiterbarkeit:  
  Jeder Decorator repräsentiert genau eine Funktion. Dadurch wird der Code leichter wart- und erweiterbar.

## Nachteile

* Erhöhte Klassenanzahl
* Erschwerte Fehlerfindung:  
  Bei langen Aufrufketten von dekorierenden Objekten sind Fehler schwer zu finden.

# Anwendungsfälle

Das Decorator-Pattern findet in folgenden Fällen Anwendung:

* Dynamische Funktionserweiterung von Objekten
* Beispiele:
  + Autos mit verschiedenen Features (Radio, Felgen, Boardcomputer, usw.)
  + Beim Speichern von Textdateien kann z.B. ein Maskieren von Umlauten oder ein Komprimierungsalgorithmus hinzugeschaltet werden
  + Stream-Klassen der Java-Bibliothek
  + GUI-Komponente verschönern (z.B. Scrollbars/Rahmen zum Textfeld hinzufügen)
  + usw.