Sommersemester 2021

Wolfgang Berger

Software Paradigmen

Structural Patterns

Strukturmuster

Strukturmuster

- Klassenbasierte Strukturmuster benutzen Vererbung, um Schnittstellen und Implementierungen zusammenzuführen
  - Klassenbasierter Adapter

Strukturmuster

- Objektbasierte Strukturmuster führen Objekte zusammen um so neue Funktionalität zu gewinnen. So können zur Laufzeit neue Objektkompositionen erstellt werden.
  - Objektbasierter Adapter
  - Bridge
  - Decorator
  - Facade
  - Flyweight
  - Proxy
  - Composite

Strukturmuster

- Composite
- Proxy
- Flyweight
- Adapter
- Bridge
- Facade
- Decorator

Composite

- Zweck
  - Füge Objekte zu Baumstrukturen zusammen, um Teil-Ganzes-Hierarchien zu repräsentieren.
  - Das Composite Pattern ermöglicht es Klienten, sowohl einzelne Objekte, als auch Kompositionen von Objekten einheitlich zu behandeln.

Composite

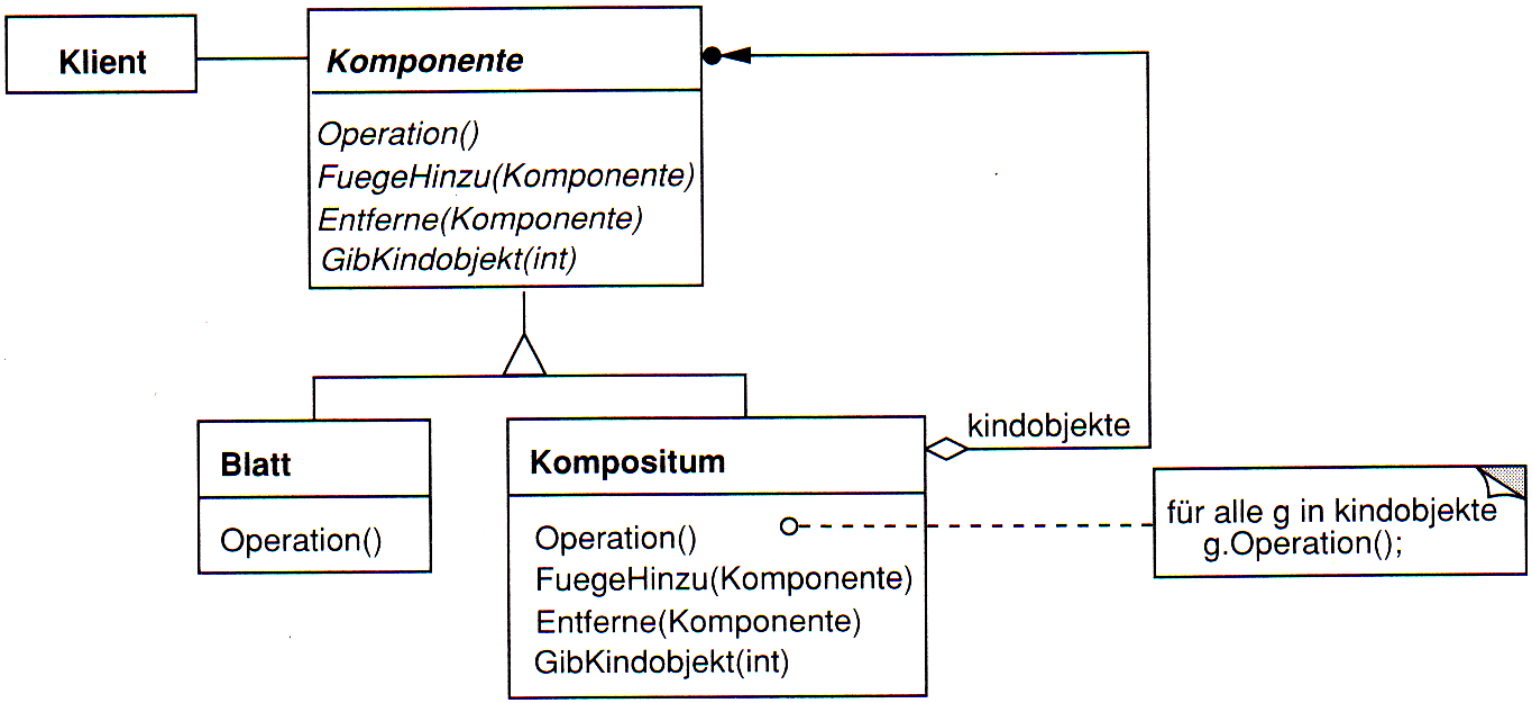
- Motivation
  - Komposition einfacher grafischer Objekte zu komplexen Ganzen
  - Gleiche Behandlung komplexer Kompositionen wie einfache Objekte
  - Keine Unterscheidung notwendig
  - Rekursive Komposition
  - Grafikanwendungen: Behälter sind Kompositionen
  - Aggregationen (bestehen aus…)

Composite

- Anwendbarkeit
  - Verwende das Kompositionsmuster, wenn,
    - Teil-Ganzes-Hierarchien von Objekten aufgebaut werden soll
    - Klienten in der Lage sein sollen, die Unterschiede zwischen zusammengesetzten und einzelnen Objekten zu ignorieren. Klienten behandeln alle Objekte in der zusammengesetzten Struktur einheitlich.

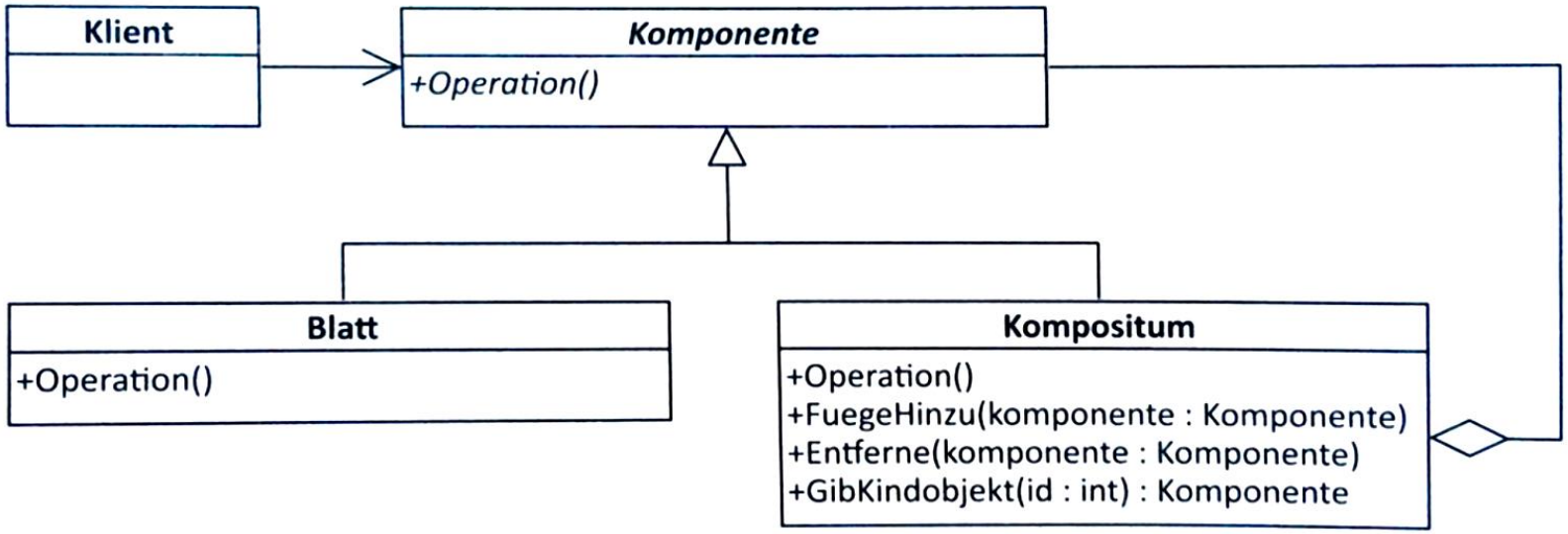
Composite

- Struktur



Composite

- Alternative Struktur



Composite

- Teilnehmer
  - Komponente
    - deklariert die Schnittstelle für Objekte in der zusammengefügten Struktur.
    - implementiert, sofern angebracht, ein Defaultverhalten für die allen Klassen gemeinsame Schnittstelle.
    - deklariert eine Schnittstelle zum Zugriff auf und zur Verwaltung von Kindobjektkomponenten.
    - definiert optional eine Schnittstelle zum Zugriff auf das Elternobjekt einer Komponente innerhalb der rekursiven Struktur und implementiert sie, falls dies angebracht erscheint.

Composite

- Teilnehmer
  - Kompositum
    - definiert Verhalten für Komponenten, die Kindobjekte haben können.
    - speichert Kindobjektkomponenten.
    - implementiert kindobjekt-bezogene Operationen der Schnittstelle von Komponente.

Composite

- Teilnehmer
  - Blatt
    - repräsentiert Blattobjekte in der Komposition. Ein Blatt besitzt keine Kindobjekte.
    - definiert Verhalten für die primitiven Objekte in der Komposition
  - Klient
    - manipuliert die Objekte in der Komposition durch die Schnittstelle von Komponente.

Composite

- Interaktionen
  - Klienten verwenden die Klassenschnittstelle von Komponente, um mit Objekten in der Kompositionsstruktur zu interagieren.
  - Wenn der Empfänger ein Blatt ist, wird die Anfrage direkt abgehandelt.
  - Wenn der Empfänger ein Kompositum ist, leitet es zumeist die Anfrage an seine Kindkomponenten weiter.

Composite

- Konsequenzen
  - Primitive Objekte ergeben in Summe ein Komplexes.
  - Komplexe Objekte können wiederum durch Rekursion zu noch komplexere Objekte zusammengesetzt werden.
  - Einheitliche Behandlung komplexer und einfacher Objekte durch den Klienten
  - Neue Blattklassen passen automatisch zu existierenden Strukturen
  - Nachteil: Typprüfung von Blattobjekten muss selbst erledigt werden (z.B. ein Kompositum, das nur aus bestimmten Objekten bestehen darf).

Composite

- Implementierung
  - Explizite Referenzen auf das Elternobjekt für leichtere Traversierung in Komponente.
  - Gemeinsame Nutzung von Komponenten – ev. mehrere Elternreferenzen. Vermindert den Speicherverbrauch.
  - Maximierung der Komponentenschnittstelle
  - Verwaltung der Kindobjekte in der Kompositumklasse
  - Ordnung der Kindobjekte manchmal wichtig

Composite

- Implementierung
  - Laufzeitverhalten beim Traversieren der Kindobjekte (z.B. Cachingalgorithmen in den Kompositumklassen)
  - Löschen von Kindobjekten durch Kompositum – vor allem bei Sprachen ohne automatische Speicherbereinigung
  - Datenstruktur zum Speichern von Kindobjekten von gewünschter Effizienz abhängig.

Composite

- Übung 4!