DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

DEKORIERER

Lars Briem

(briem.lars@googlemail.com)

Duale Hochschule Baden Württemberg - Standort Karlsruhe

Dekorierer

- Dynamische Zuweisung einer weiteren Verantwortung bzw. Zuständigkeit zu einem Objekt
- Flexible Alternative für Objekthierarchien

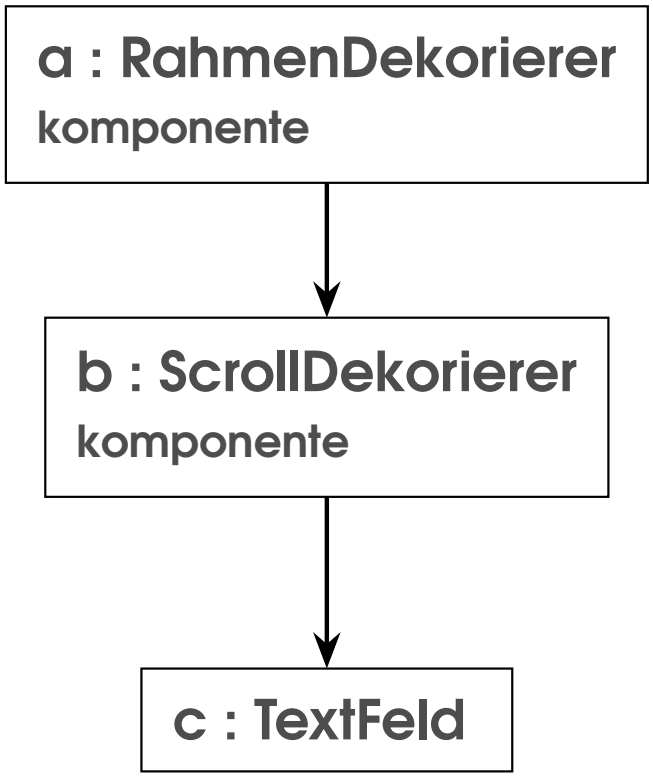
Dekorierer – Einordnung

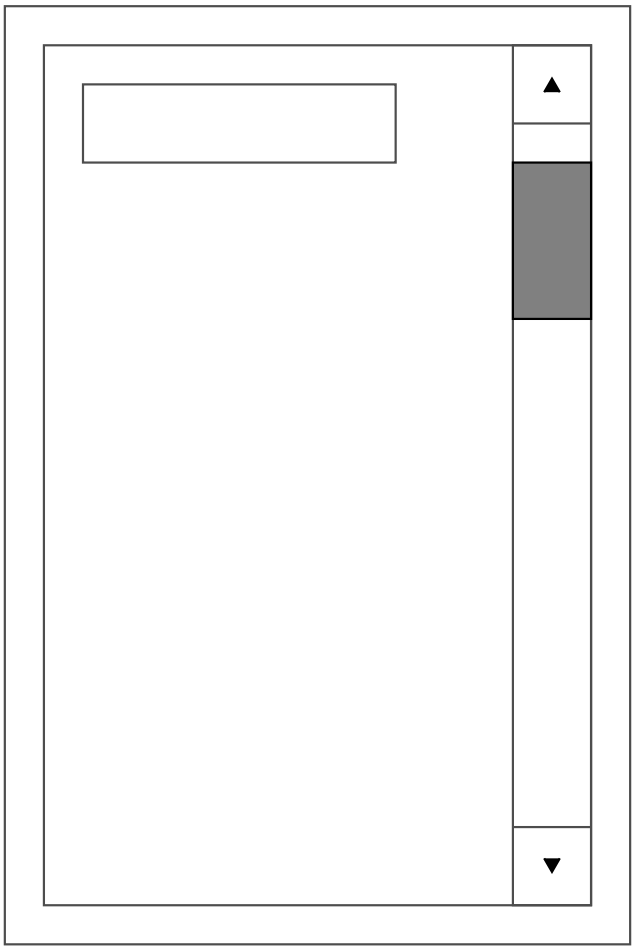
- Objektbasiertes Strukturmuster
- Leichtgewichtig
- Instanzenreich
- Auch bekannt als
  - Decorator
  - Wrapper

Dekorierer – Motivation

- Das Hinzufügen von Zuständigkeiten zu einer Klasse mittels Ableitung ist sehr starr
  - Anwender hat keine Entscheidungsgewalt
- Verschachtelung von Objekten zum Hinzufügen von Funktionalität liefert mehr Freiheiten bzw Kombinationsmöglichkeiten
- Zusatzfunktionalität soll transparent dazwischen liegen

Dekorierer – Beispiel

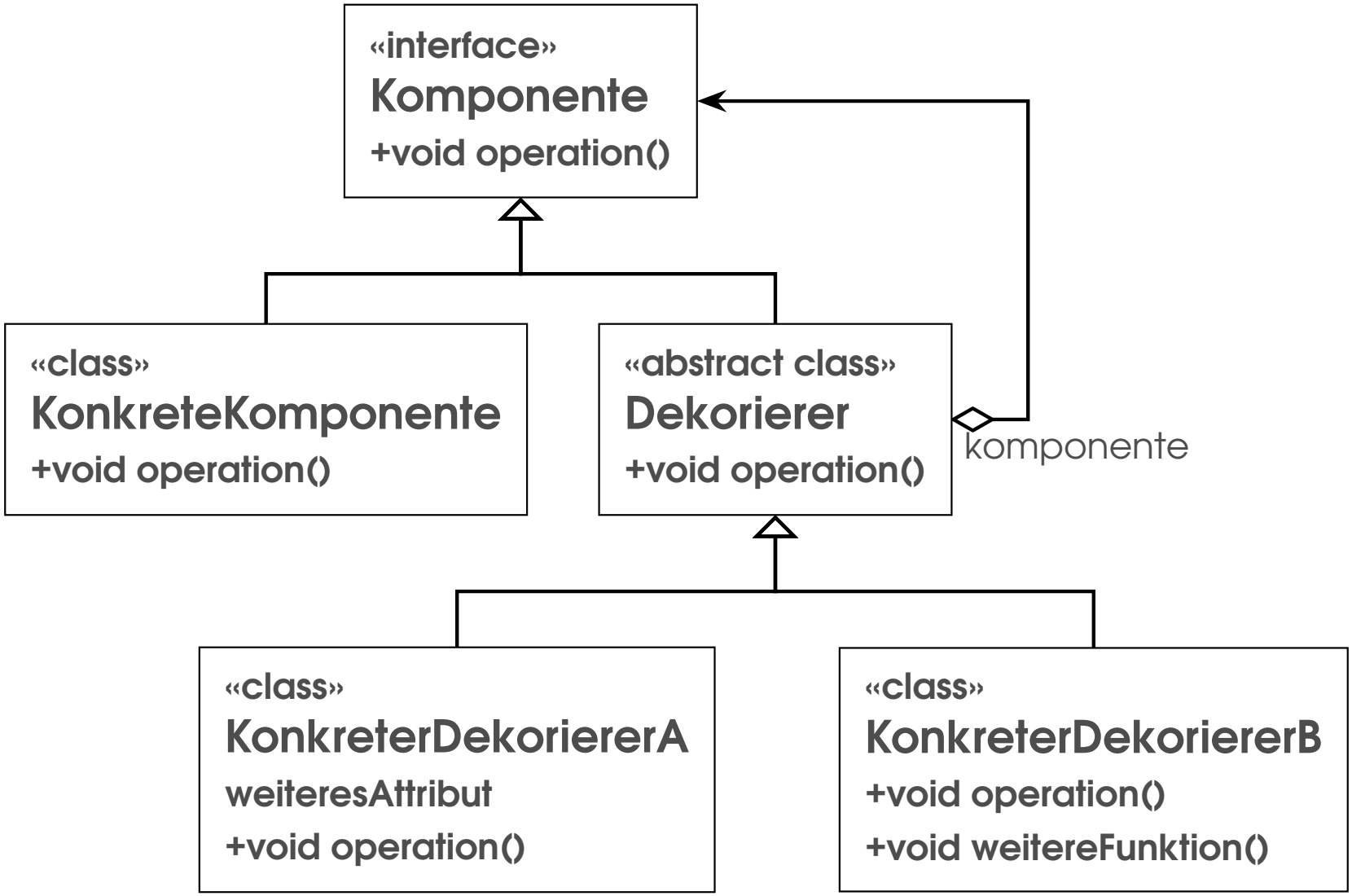




Dekorierer – Anwendung

- Zuweisung von Zuständigkeiten zu einzelnen Objekten dynamisch und transparent, ohne andere zu beeinflussen
- Für entfernbare Zuständigkeiten
- Wenn Ableitung einer bestehenden Klasse zu komplex ist bzw. die Objekthierarchie extrem aufgebläht wird

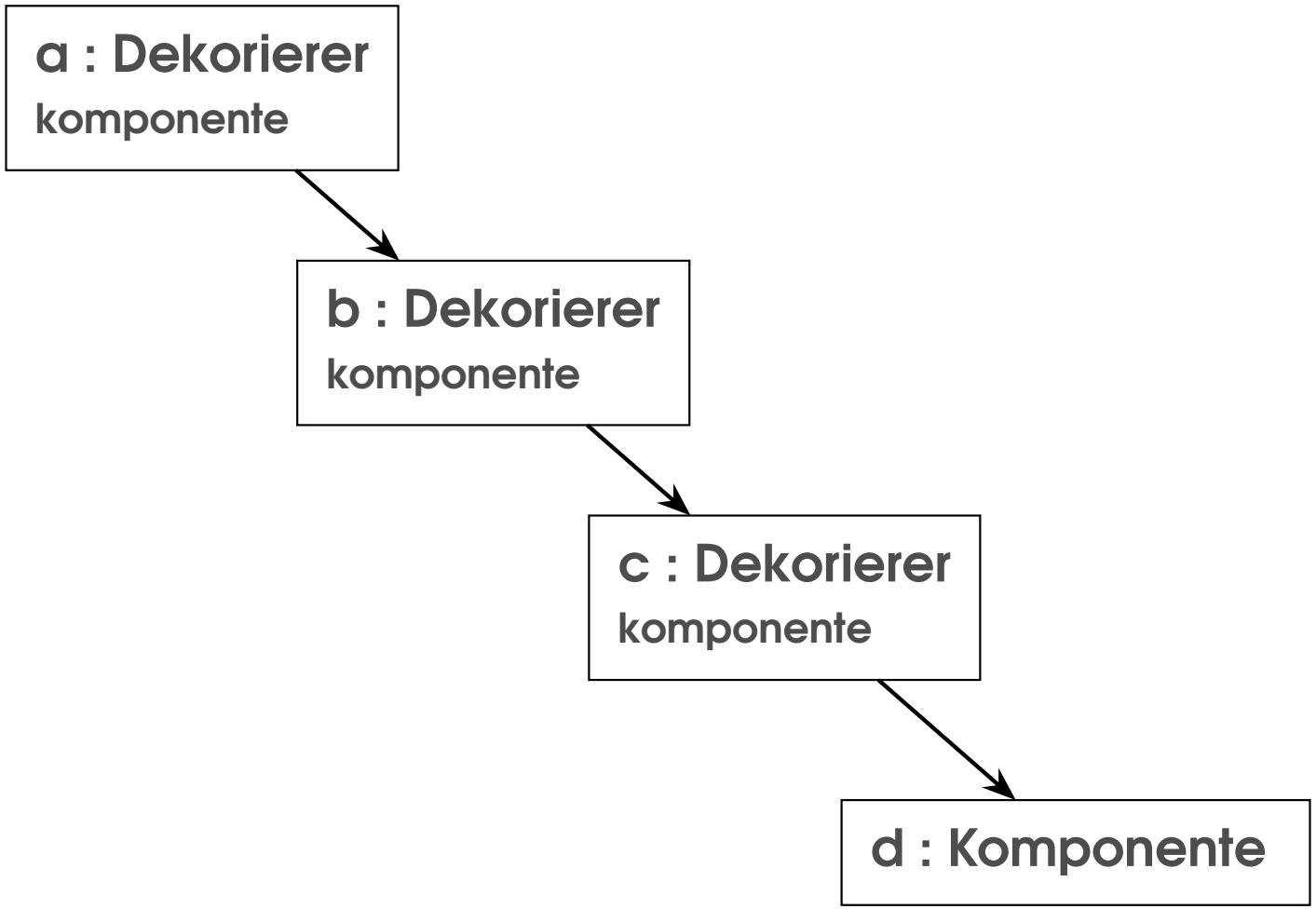
Dekorierer – Struktur



Dekorierer – Akteure

- Komponente
  - Definiert das Interface, das dynamisch erweitert werden soll
- Konkrete Komponente
  - Definiert Komponente, die dynamisch erweitert werden kann
- Dekorierer
  - Hält eine Referenz auf eine Komponente
  - Implementiert das Interface der Komponente
- Konkreter Dekorierer
  - Fügt weitere Zuständigkeit zur Komponente hinzu

Dekorierer – Interaktion der Akteure



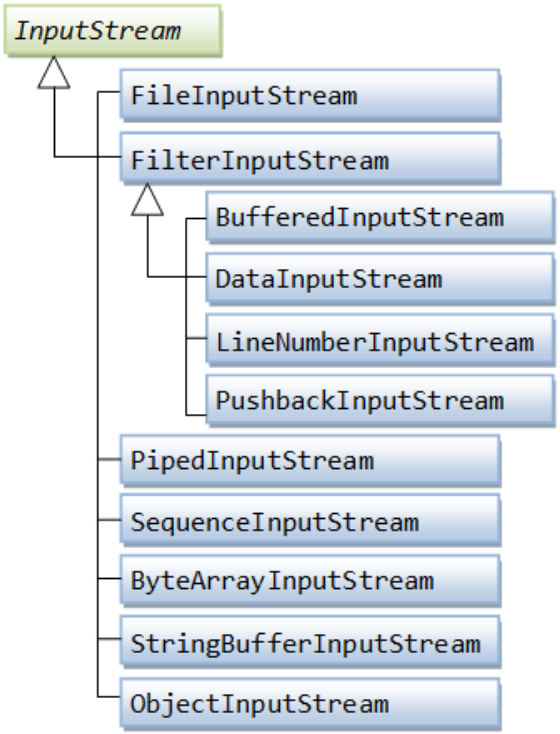
Dekorierer – Auswirkungen

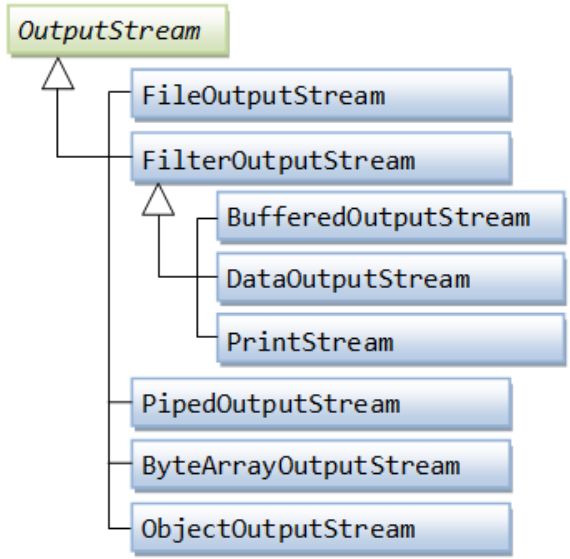
+ Flexiblere Alternative zur Ableitung von Objekten
  + Zuständigkeit kann dynamisch hinzugefügt bzw. entfernt werden
  + Beliebige Kombination von Zuständigkeiten, auch mehrfach
+ Führt zu einfachen, zusammensteckbaren Klassen
  + Unterstützt das Open Closed Principle
+ Vermeidet große konfigurierbare Klassen

Dekorierer – Auswirkungen

- Identität des Dekorierers und der Komponente unterschiedlich
  - `equals` und `hashCode` liefern für Dekorierer und Komponente `false`
- Viele kleine Objekte erschweren das Debuggen bzw. Lernen des Systems für ungeübte

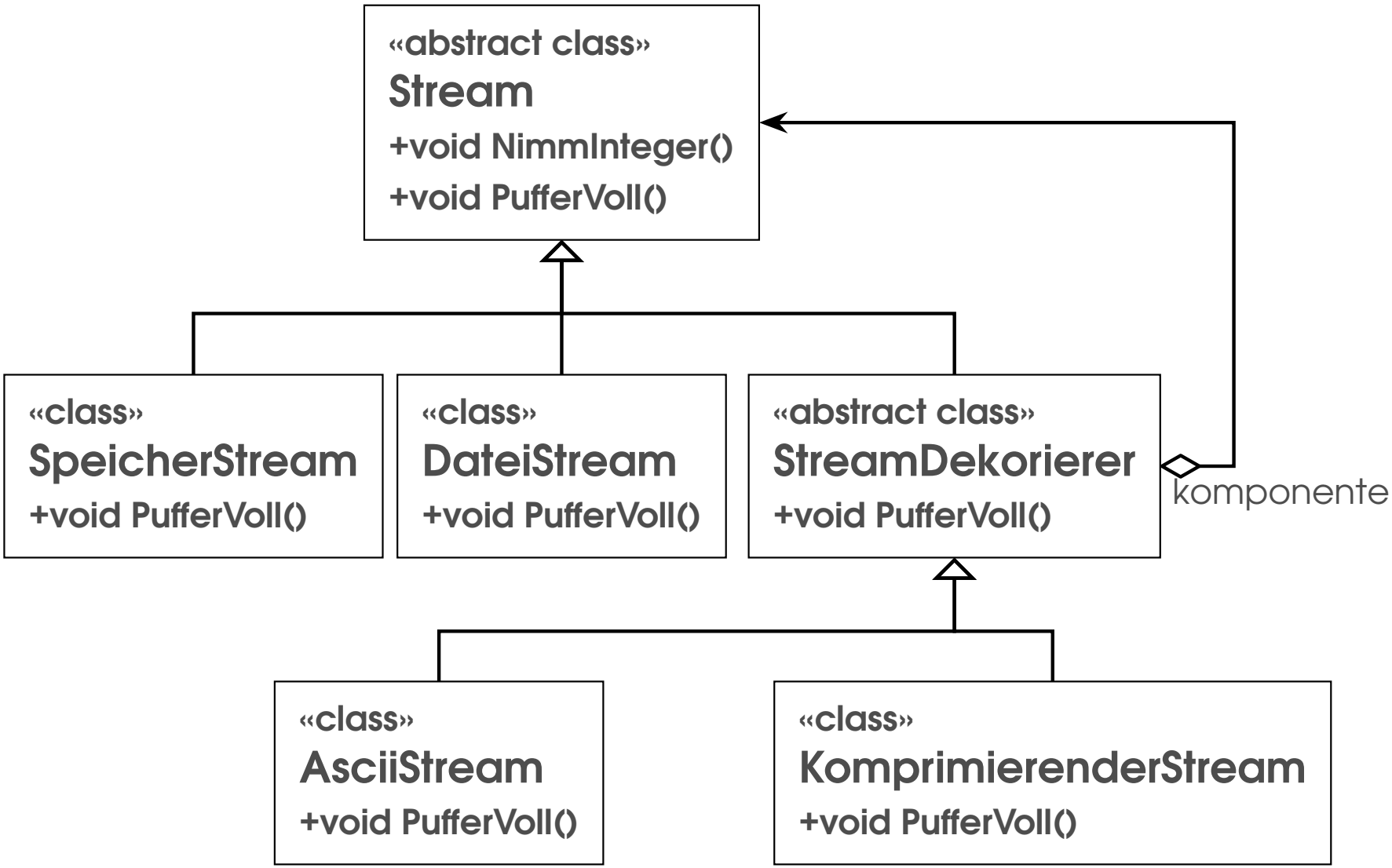
Dekorierer – Beispiel - InputStream






```
InputStream input = new AudioInputStream(format,
                    new DecryptInputStream(secret,
                    new UncompressInputStream(zipType,
                    new BufferedInputStream(
                    new FileInputStream(someFile)
                    ))));
```

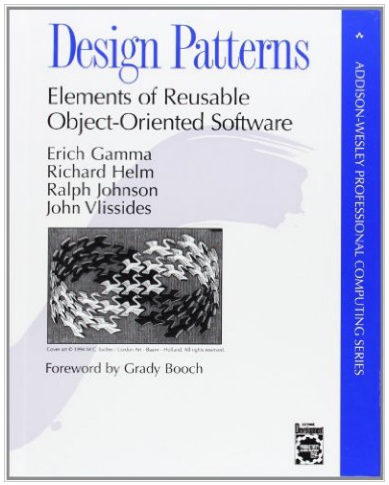
Dekorierer – Beispiel - Streamstruktur



Dekorierer – Zusammenfassung

- Erweitern eines Objekts mit zusätzlicher Funktionalität
- Einhaltung einer flachen Objekt-Hierarchie
- Zusätzliche Funktionalität bleibt transparent

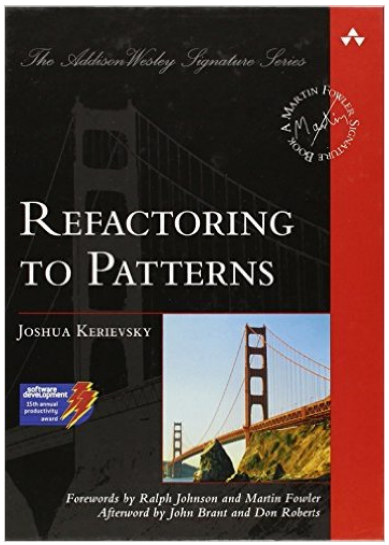
Literatur



- Design Patterns
  - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
  - Addison-Wesley
  - ISBN: 978-0201633610

Weitere Infos

- Entwurfsmuster auf YouTube
  - John Lindquist erklärt Entwurfsmuster mit StarCraft II
  - `https://www.youtube.com/playlist?list=PL8B19C3040F6381A2`



- Refactoring to Patterns
  - Joshua Kerievsky
  - Addison-Weslay
  - ISBN: 978-0321213358