Universität der Bundeswehr
Fakultät für Informatik – Institut 2
Prof. Dr.-Ing. Mark Minas

FT 2025
Übung 8

Objektorientierte Programmierung

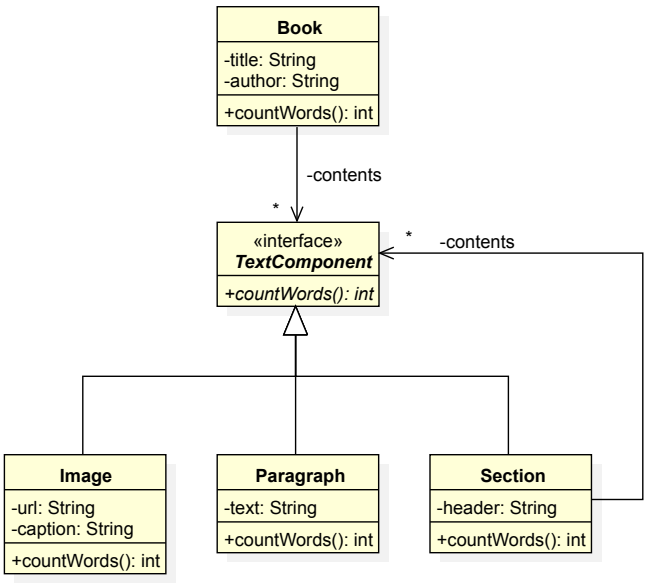
02.06.2025

Lernziele: Composite- und Visitor-Pattern

Allgemeiner Hinweis: Es wird erwartet, dass Sie die Aufgaben bereits **vor** der Übung gelöst haben und Sie die Lösung vor Ort präsentieren können.

Aufgabe 1 (Composite Pattern)

In dieser Aufgabe greifen wir das in der Vorlesung im Zusammenhang mit dem Composite Pattern genutzte Beispiel auf, in dem es um Bücher geht. Konkret geht es hier um Bücher, die aus einzelnen Kapiteln, Abschnitten, Unterabschnitten etc. bestehen, und natürlich aus Textabschnitten und Abbildungen. Das folgende Klassendiagramm stellt die in dieser Übung verwendeten Klassen und Interfaces dar:



Dabei spielen das Interface `TextComponent` die Rolle des Components des Composite Patterns, `Image` und `Paragraph` die *Leaf*-Rolle, und `Section` die Rolle des *Composites*. Die Klasse `Book` hat einen Autor sowie einen Titel und besteht aus einer Folge von Textkomponenten.

a) Laden Sie aus dem Ilias-Bereich die Datei `uebung08-src.zip` herunter und legen Sie den Inhalt in einem neuen Projekt ab. Die enthaltene Klasse `BookDemo` enthält Code, der ein Buch mit den im Folgenden zu realisierenden Klassen und Interfaces instanziiert und noch zu schreibende Methoden aufruft. Die Datei lässt sich natürlich noch nicht ohne Fehler übersetzen, bevor Sie die folgenden Teilaufgaben bearbeitet haben.

b) Erstellen Sie im Package `doc` die im Klassendiagramm angegebenen Klassen und Interfaces. Die notwendigen Konstruktoren können der Datei `BookDemo.java` entnommen werden. Die Methoden `countWords()` schreiben Sie in der nächsten Teilaufgabe.

c) Schreiben Sie nun die Methode `countWords()` in jeder Klasse. Aufgerufen für ein Buch soll sie die Anzahl aller Wörter zählen und zurückgeben, die in Textabschnitten vom Typ `Paragraph` vorkommen. Ignoriert werden sollen Überschriften in Abbildungen und Abschnitten etc. vom Typ `Section`. Wenden Sie dabei die für das Composite Pattern charakteristische Vorgehensweise an.

Hinweis: Wir bezeichnen in dieser Aufgabe jede Teilzeichenfolge als Wort, die aus alphabetischen Zeichen besteht und von nicht-alphabetischen Zeichen (bzw. Anfang oder Ende der Zeichenkette) begrenzt wird. Konsultieren Sie die Dokumentation der Klassen `String` und `Character`, um herauszufinden, wie man auf die Buchstaben eines Strings zugreifen und wie man prüfen kann, ob ein Zeichen alphabetisch ist.

Aufgabe 2 (Visitor Pattern)

In dieser Aufgabe soll die Klasse `Book` zusätzliche Methoden erhalten, die u.a. ein Inhaltsverzeichnis erstellen und das gesamte Buch als einen zusammenhängenden String ausgeben. Für jede dieser Methoden müsste man entsprechende Methoden in jeder der oben realisierten Klassen und Interfaces schreiben. Um dies zu vermeiden und die Lösung besser zu strukturieren, sollen Sie das Visitor Pattern einsetzen.

a) Ergänzen Sie im Interface `TextComponent` eine `accept`-Methode, die Sie in `Image`, `Paragraph` und `Section` geeignet implementieren. Hierfür benötigen Sie zusätzlich ein entsprechendes `Visitor`-Interface.

b) In dieser Teilaufgabe sollen Sie im Grunde Aufgabe 1c noch einmal lösen, diesmal aber mit Hilfe eines Visitors. Erstellen Sie dazu eine Klasse `CountWordsVisitor`, die `Visitor<Integer>` implementiert und die in Analogie zu Aufgabe 1c die Wörter zählt, die in `TextComponent`-Mitgliedern vorkommen. Ergänzen Sie in der Klasse `Book` eine Methode `int countWordsByVisitor()`, die eine Instanz von `CountWordsVisitor` verwendet, um die Zahl der im Buch vorkommenden Wörter zu ermitteln. Das Ergebnis soll natürlich mit dem der Methode `countWords()` übereinstimmen.

   Vergleichen Sie diese Lösung mit der in Aufgabe 1c.

c) Erstellen Sie eine Klasse `TableOfContentsVisitor`, die `Visitor<List<String>>` implementiert und eine Liste der Überschriften in einem `TextComponent`-Mitglied erstellt. Dabei sollen die Überschriften numeriert werden. Überschriften von Sections auf oberster Ebene (die üblicherweise als Kapitel bezeichnet werden) erhalten die Nummern 1, 2, etc. Darin enthaltene Sections (die Üblicherweise Abschnitte genannt werden) bekommen die Nummern 1.1, 1.2, etc., wenn sie im Kapitel 1 enthalten sind, etc.

   Ergänzen Sie in der Klasse `Book` ferner eine Methode `List<String> tableOfContents()`, die eine Instanz von `TableOfContentsVisitor` verwendet, um das Inhaltsverzeichnis in Form einer Liste von numerierten Überschriften zu bestimmen. Für das in `BookDemo.java` erstellte Buch soll folgende Liste zurückgegeben werden:

```
[1 Einführung, 2 Hintergrund, 2.1 Geschichte, 2.2 Literatur, 3 Idee,
 4 Umsetzung, 4.1 Entwurf, 4.1.1 Grobentwurf, 4.1.2 Feinentwurf,
 4.2 Realisierung, 5 Zusammenfassung und Ausblick]
```

d) Erstellen Sie eine Klasse `ToTextVisitor`, die `Visitor<String>` implementiert und beliebige `TextComponent`-Mitglieder als Text darstellt. Dabei sollen die Überschriften wie in der vorigen Teilaufgabe und auch die Abbildungsunterschriften („captions“) numeriert werden. Auch die Abbildungen sollen innerhalb ihrer enthaltenen Section sukzessive nummeriert werden (siehe unten). Ergänzen Sie in der Klasse `Book` ferner eine Methode `String toText()`, die eine Instanz von `ToTextVisitor` verwendet, um das Buch in Textform zurückzugeben.

   Nachstehend ist ein Teil der Ausgabe für das in `BookDemo.java` erstellte Buch angegeben. Dem Beispiel ist zu entnehmen, wie Abbildungsunterschriften zu numerieren sind.

```
  Peter Mustermann
Die eierlegende Wollmilchsau

Über Entwurf und Realisierung eierlegender Wollmilchsäue.
```

1 Einführung

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam ...

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse ...

2 Hintergrund

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper...

2.1 Geschichte

Nam liber tempor cum soluta nobis eleifend option congue nihil...

<image https://www.bekannt.de/p1.pdf>
Fig. 2.1.1: Bild 1

Duis autem vel eum iriure dolor in hendrerit in vulputate velit ...

2.2 Literatur

At vero eos et accusam et justo duo dolores et ea rebum. Stet ...

<image https://www.eh-klar.de/p2.pdf>
Fig. 2.2.1: Bild 2

Consetetur sadipscing elitr, sed diam nonumy eirmod tempor ...

<image https://www.jeder-weiss-es.de/p3.pdf>
Fig. 2.2.2: Bild 3