Übungen zur Vorlesung

Softwaretechnologie

- Wintersemester 2015/16 -

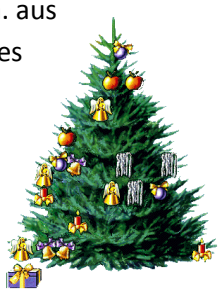
Dr. Günter Kniesel

Übungsblatt 10

Zu bearbeiten bis: 22.01.2016

Bitte fangen Sie **frühzeitig** mit der Bearbeitung an, damit wir Ihnen bei Bedarf helfen können. Checken Sie die Lösungen zu den Aufgaben in Ihr Repository ein, „Erklärungen" bitte als Textdatei. Fragen zu Übungsaufgaben/Vorlesung können Sie auf der Mailingliste swt-tutoren@lists.iai.uni-bonn.de, bzw. swt-vorlesung@lists.iai.uni-bonn.de stellen.

Aufgabe 1. *Jahreszeitbedingte Anwendung von Entwurfsmustern* (16 Punkte)



Ein reichlich geschmückter Weihnachtsbaum besteht ausschließlich aus *Baumbestandteilen*, d.h. aus *Zweigen*[1] (an denen weitere immer feiner verästelte Zweige wachsen können) und am Ende eines Zweiges evtl. *Baumschmuck*. Baumschmuck können *Engel*, *Kugeln*, *Sterne,* oder *Kerzen* sein.

a) Modellieren Sie den Weihnachtsbaum in Java unter Zuhilfenahme eines geeigneten Entwurfsmusters. Annotieren Sie dabei die Elemente (Klassen, Methoden, Felder) Ihres Modells mit den Rollen die sie im Entwurfsmuster spielen (Java-Annotationen).

b) Entwickeln Sie ein Programm, das Ausgehend vom Stamm einen zufällig generierten Weihnachtsbaum mit verschiedenem Baumschmuck erzeugt. Achten Sie dabei darauf, dass in unserem Kulturkreis die maximale Zweigrekursionstiefe 3 beträgt und jeder Zweig maximal 5 Kindelemente haben kann. Geben Sie eine Beschreibung ihres Baumes auf der Konsole aus, indem Sie die *toString()*-Methoden der Baumelemente geeignet implementieren und auf dem Stamm aufrufen.

c) Zur Baumpflege werden im Rheinland traditionell Heinzelmännchen eingesetzt, kleine Spezialisten, die den Baum entlang klettern und eine spezifische Aktion auf den konkreten Baumelementen ausführen können. Modellieren Sie in Ihrem Projekt unter Nutzung eines geeigneten Entwurfsmusters ein abstraktes *Heinzelmännchen*. Ändern Sie, wenn nötig auch bestehende Klassen, so dass später beliebige konkrete Heinzelmännchen die Baumpflege übernehmen können. Beachten Sie, dass die Ausprägung einer konkreten Heinzelmännchen-Aktion je nach Baumelement unterschiedlich sein kann! Annotieren Sie die am Entwurfsmuster beteiligten Klassen und Methoden mit ihren Rollen, wie in (a).

d) Implementieren Sie zwei konkrete Heinzelmännchen:

- *Kugel-Anmal-Heinzelmännchen*: Setzt die Farbe von Kugeln auf „rot".
- *Kerzen-Anzünde-Heinzelmännchen*: Setzt den Status jeder Kerze auf „brennend".

Ergänzen Sie evtl. zusätzlich benötigte Eigenschaften in den konkreten Baumelementen.

[1] Der Einfachheit halber sehen wir Nadeln als integralen Bestandteil der Zweige an und modellieren sie hier nicht. Des Weiteren handelt es sich beim Stamm des Baumes auch um einen (überdimensionierten) Zweig.

Aufgabe 2. *Umsetzung von Modellen (2+8+2+3 Punkte)*

In dieser Aufgabe stellen wir Ihnen ein Klassendiagramm und Sequenzdiagramm zur Verfügung, das einen Taschenrechner mit Plugin-Fähigkeiten modelliert. Ihre Aufgabe liegt darin das gegebene Modell in Java umzusetzen.

Die benötigten Diagramme finden Sie unter:
ssh://git-se@git.iai.uni-bonn.de/swt2015_readonly

Die blau eingefärbten Elemente des Klassendiagrammes geben wir Ihnen vor. Sie sind als Java-Code ebenfalls im readonly-ordner enthalten.

a) Analysieren Sie das gegebene Klassendiagramm. Identifizieren Sie welche Entwurfsmuster in diesem Modell verwendet wurden.

b) Setzen Sie das Modell in Java um. Verwenden Sie das Klassendiagramm um die Struktur, das Sequenzdiagramm um den Ablauf einer Rechenaktion zu verstehen. Beachten Sie zusätzlich die folgenden Informationen:

- Alle Operationen (Add, Sqrt,…) sollen als .class Dateien vom OperatorLoader geladen werden, und sind nicht Teil des Rechners.
- Verwenden Sie die gegebenen *Add.class* und *PrimeDecomposition*.class Dateien, um ihre Implementierung zu testen. (Token: Add (*+*), PrimeDecomposition (*prime*))
- Der Einfachhalt halber können sie die Präfix Notation für Eingaben verwenden (z.B.: „+ 2 3“ oder „prime 18“).
- Ergänzen Sie eigene Methoden, wenn es ihnen notwendig erscheint. Achten Sie in diesem Fall darauf, dass Sie Modell und Code konsistent halten.

c) Ergänzen Sie eine neue Operation, die ebenfalls als Class-Datei vom OperatorLoader geladen werden kann.

d) Erklären Sie wie Sie die verschiedenen Elemente des Klassendiagramms in Code umgesetzt haben.