Einführung in Software Engineering WS 10/11

Fachbereich Informatik

Dr. Michael Eichberg
eichberg@informatik.tu-darmstadt.de

Assistent: Ralf Mitschke
mitschke@st.informatik.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT

Zusatz Übungsblatt: Command Pattern

Freiwillige Bearbeitung! Diese Übung können Sie zum weiteren Selbststudium zum Thema Design Patterns bearbeiten. Es wird keine Bonuspunkte auf die Aufgaben in dieser Übung geben. Die Aufgaben werden nicht durch die Tutoren korrigiert. Falls Sie Feedback auf Ihre Lösung erhalten möchten, können Sie dieses in den Sprechstunden der Veranstalter erhalten. Bringen Sie bitte Ihre Lösungen ausgedruckt mit (ausgenommen Code) und halten Sie ein einen Laptop mit Ihrem Source Code bereit.

Aufgabe 1

Ziel: *Anwendung des Command Patterns*

Die Flashcards-Anwendung soll so erweitert werden, so dass das UI weniger Kopplung zu den Operationen des Domänenmodells aufweist.

a) Entkopplung von Domänenmodell Operationen

Schreiben Sie Klassen, die die Folgenden drei Operationen mittels des Command Patterns implementieren:

1. Hinzufügen einer Flashcard
2. Entfernen einer Flashcard
3. Änderungen am Inhalt (Frage- und/oder Antwortseite) einer Flashcard

Schreiben Sie für jedes der drei Commands Unit-Tests, die prüfen, ob die Operation erfolgreich durchgeführt wurde.

Nutzen Sie die erstellten Commands, um die bisherigen Aufrufe für die drei Operationen in den Klassen `FlashcardsWindow` und `FlashcardEditor` zu ersetzten. Diese UI Klassen, können danach immer noch die Klasse `Flashcard` oder `FlashcardSeries` nutzen. Es dürfen aber keine Abhängigkeiten zu den konkreten Methoden für die drei implementierten Operationen (z.B. `Flashcard.setQuestion(String text)` oder `FlashcardSeries.addCard(Flashcard flashcard)`) bestehen. Andere Methoden, wie z.B. Methoden zum Setzten der Metadaten, sind nicht Teil des Command Patterns und können weiter genutzt werden.

b) Dokumentation des Command Patterns

Erstellen Sie ein UML Klassendiagramm welches Ihre Implementierung des Command Patterns dokumentiert. Die Klassen sollten alle für das Pattern relevanten Methoden zeigen. Weitere Methoden sind nicht aufzuführen. Notieren Sie welche Klassen welche Rollen des Patterns inne haben (entweder im Diagramm, siehe „Design Pattern – Introduction“ Folie 27, oder extern in Ihrem Lösungsdokument).

Erläutern Sie kurz welche Klassen in Ihrer Implementierung für das Erstellen eines Commands Verantwortlich sind. Welche Klassen haben dadurch Kopplung zu den konkreten Commands und ist dies ein gutes Design? Diskutieren Sie ggf. einen Verbesserungsvorschlag.

Aufgabe 2

Ziel: *Anwendung von Design Patterns für Undo/Redo*

Die Flashcards-Anwendung soll um eine multi-level Undo und Redo Funktionalität erweitert werden. Einmal durchgeführte Aktionen können mittels Undo widerrufen werden und die Daten der Anwendung werden wieder in den vorherigen Zustand versetzt. Es können beliebig viele Aktionen wiederrufen werden. Mittels Redo wird eine entsprechende Aktion wieder ausgeführt. Abb. 2 zeigt einen Vorschlag, wie dies in der GUI umgesetzt werden kann.

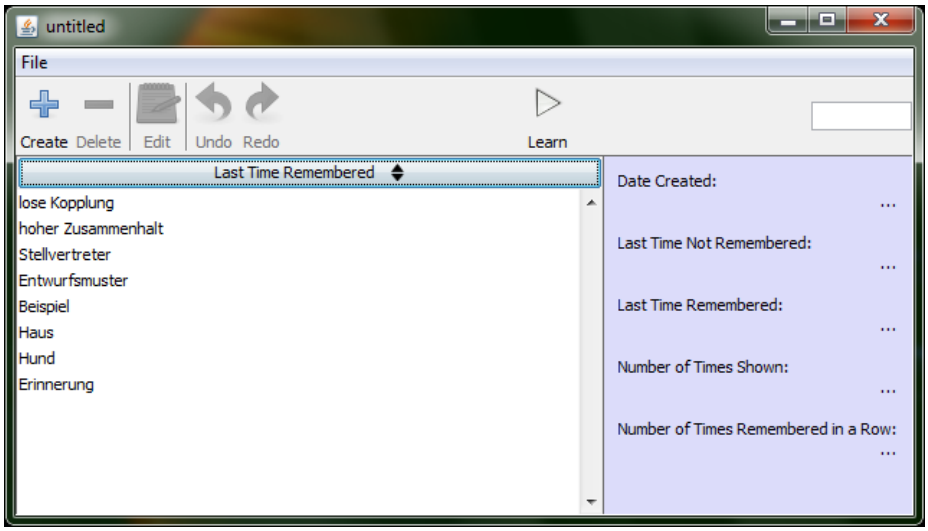

Abb. 1 Beispiel Flashcards-Anwendung mit Undo/Redo Funktion

Schreiben Sie Klassen, die es ermöglichen die drei folgenden Operationen (siehe auch Aufgabe 1) Rückgängig zu machen (Undo) und wieder herzustellen (Redo):

1. Hinzufügen einer Flashcard
2. Entfernen einer Flashcard
3. Änderungen am Inhalt (Frage- und/oder Antwortseite) einer Flashcard

Nutzen Sie hierzu Ihre Implementierung des Command Patterns aus Aufgabe 1 und erweitern Sie diese, um Undo und Redo zu erlauben. Die Buttons für Undo und Redo sollen dabei nur aktiv sein, wenn tatsächlich etwas rückgängig gemacht bzw. erneut ausgeführt werden kann. Nutzen Sie hierzu das Observer Pattern.

Schreiben Sie für jedes Command mindestens einen Test, der prüft, ob die Operation erfolgreich Rückgängig gemacht wurde, sowie einen Test, der prüft, ob die Operation erfolgreich erneut ausgeführt wurde.

Hinweis: *Zur konsistenten Speicherung des Zustands der Applikation bietet sich das Memento Pattern an (siehe „Design Patterns“ – Gamma et al.).*