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

Übungsblatt 9 (10 Punkte): Einführung in Design Patterns

Abgabeformat: Reichen Sie ihre Lösung per SVN ein. Jede Übung muss in einem eigenen Ordner ***ex<Number>*** *(<Number> = 01, 02, ...)* in Ihrem Gruppenverzeichnis eingereicht werden. Während der Übungsbearbeitung können Sie Ihre Lösungen beliebig oft in das SVN hochladen (per Commit). Wir prüfen die Zeit der Einreichung Ihrer Lösungen unter der Benutzung des SVN Zeitstempels.

Erstellen Sie für Lösungen der Aufgaben, die keinen Quelltext erfordern, eine PDF-Datei mit dem Dateinamen ***solution.pdf***. Dies gilt auch für alle UML-Diagramme, die Sie erstellen. Die Basisanwendung wird als Eclipse-Projekt vorgegeben. Ihr eigener Code muss entsprechend in den dafür vorgesehenen Verzeichnissen (***/src*** oder ***/test***) erstellt werden.

Abgabetermin: 26.01.2011 - 24:00 Uhr

Aufgabe 1 (4 Punkte)

Ziel: *Analyse der Verwendung von Design Patterns im JDK*

a) Erkennen von Design Patterns (2 Punkte)

Im Folgenden ist ein Auszug der Dokumentation der Klasse **`java.util.Arrays`** des JDKs gegeben.

Auf die Verwendung welcher(s) Patterns können Sie aufgrund der Signaturen und der Dokumentation der Methoden schließen? Welche Rolle in der Implementierung des Patterns nimmt die Klasse „Arrays“ ein? Welche Variante der Implementierung wurde gewählt?

`static int`	**`binarySearch`**`(short[] a, int fromIndex, int toIndex, int short key)` Searches a range of the specified array of shorts for the specified value using the binary search algorithm.
`static int`	**`binarySearch`**`(short[] a, short key)` Searches the specified array of shorts for the specified value using the binary search algorithm.
`static int`	**`binarySearch`**`(T[] a, int fromIndex, int toIndex, T key, Comparator<? super T> c)` Searches a range of the specified array for the specified object using the binary search algorithm.
`static <T> int`	**`binarySearch`**`(T[] a, T key, Comparator<? super T> c)` Searches the specified array for the specified object using the binary search algorithm.
`static void`	**`sort`**`(Object[] a, int fromIndex, int toIndex)` Sorts the specified array of objects into ascending order, according to the natural ordering of its elements.
`static void`	**`sort`**`(short[] a)` Sorts the specified array of shorts into ascending numerical order.
`static void`	**`sort`**`(short[] a, int fromIndex, int toIndex)` Sorts the specified range of the specified array of shorts into ascending numerical order.
`static <T> void`	**`sort`**`(T[] a, Comparator<? super T> c)` Sorts the specified array of objects according to the order induced by the specified comparator.

Der Code der Klasse `Comparator<T>` aus der Implementierung ist im Folgenden (siehe Seite 2) auszugsweise angegeben.

```
public interface Comparator<T> {
          /**
          * Compares its two arguments for order. Returns a
          * negative integer, zero, or a positive integer
          * as the first argument is less than, equal to,
          * or greater than the second.
          */
          int compare(T o1, T o2);
          …
}
```

b) Erkennen von Design Patterns (2 Punkte)

Studieren Sie die Implementierung der Klasse `java.util.AbstractCollection<E>` des Java JDKs (1.6). Welche Ihnen bekannten Patterns werden verwendet? Beschreiben Sie kurz für jedes gefundene Pattern wie es in dieser Klasse zum Einsatz kommt und erläutern Sie die Funktionsweise des Patterns anhand von Beispielmethoden aus dieser Klasse.

Aufgabe 2 (6 Punkte)

Ziel: *Erweiterung der Flashcards-Anwendung mit verschiedenen Lernstrategien*

Um das Lernen von Karteikarten effektiv zu unterstützen, soll die Flashcards-Anwendung verschiedene Lernstrategien bereitstellen. Eine Lernstrategie ist dafür zuständig die Karten beim Lernen in wohl definierter Reihenfolge zu präsentieren. Dies entspricht der **User-Story 3** aus Übung 7.

Nach dem Klicken auf den „Learn Button“ (siehe Abb. 1) soll dem Anwender ein Dialog präsentiert werden, der es ermöglicht eine Lernstrategie auszuwählen (siehe Abb. 2). Anschließend werden die Karten in der durch die Lernstrategie vorgegebenen Reihenfolge im Lerndialog gelernt.

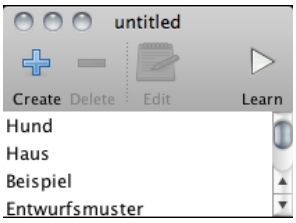

Abbildung 1: Die Flashcards Anwendung

Abbildung 2: Auswahldialog zur Lernstrategie

Die Anwendung ist so anzupassen, dass die Menge der Lernstrategien auch später einfach erweiterbar ist. Nutzen Sie daher das „Strategy“ Design Pattern, um Lernstrategien in Ihrer Flashcards-Anwendung umzusetzen. Planen Sie Ihre Umsetzung der Aufgabe anhand der Folien zum „Strategy“ Design Pattern.

Beachten Sie bei der Implementierung der Lernstrategien eine saubere Einhaltung der Trennung zwischen GUI-Logik und Domänenlogik.

a) Einfache Strategien (1 Punkt)

Implementieren Sie mindestens 2 der folgenden Lernstrategien und testen Sie Ihre Implementierung mittels JUnit Tests. Die hier aufgeführten Strategien erfordern keine Erweiterung des Domänenmodels.

- mit der neuesten Karten beginnend
  (D.h. beginnend mit der Karte, die zuletzt hinzugefügt wurde)
- mit der ältesten Karte beginnend
  (D.h. beginnend mit der Karte, die als erstes hinzugefügt wurde)
- zufällige Reihenfolge und jede Karte genau einmal
- zufällig, bis zum manuellen Abbruch

b) Erweiterte Strategien (3 Punkte)

Implementieren Sie die folgenden Lernstrategien und testen Sie Ihre Implementierung mittels JUnit Tests. Die hier aufgeführten Strategien erfordern eine Erweiterung des Domänenmodels, um zu erkennen wann und wie oft eine Karte gelernt wurde.

- „Quiz“, hier werden nur bereits gelernte Karten herangezogen.
- „Just New“, hier werden nur Karten, die noch nie gelernt wurden herangezogen.
- „Systematic“, dies ist eine lernpsychologisch sinnvolle Reihenfolge. Die Karten werden in fünf virtuelle Fächer eingeteilt. Neue Karten kommen in das vorderste Fach. Immer wenn eine Karte erfolgreich gelernt wurde, wird diese in das nächste Fach einsortiert. Innerhalb der Fächer sind die Karten danach sortiert, wann man sich das letzte Mal an die Antwort erinnert hat.
  Die „Systematic“ Strategie geht alle Fächer sequentiell durch. Es muss keine Auswahl der Fächer stattfinden.

c) Dokumentieren des Designs (2 Punkte)

Erstellen Sie ein UML Sequenzdiagramm, das verdeutlicht wie das „Strategy“ Pattern in Ihrem Projekt zum Einsatz kommt. Das Sequenzdiagramm soll exemplarisch anhand einer konkreten Lernstrategie die folgenden Abläufe aufzeigen:

- Konfiguration des Kontextes mit einer Lernstrategie
- Auswahl einer Karte mittels einer Lernstrategie aus dem Kontext heraus. Der konkrete Algorithmus der Strategie muss nicht veranschaulicht werden.

Hinweis: *Diese Aufgabe ist aufwendiger und erfordert die Implementierung mehrerer neuer Klassen. Darüber hinaus ist es erforderlich die bestehende Klasse „LearnDialog“ anzupassen.*

Hinweis: *Um einen Dialog anzuzeigen, der die Auswahl verschiedener Strategien ermöglicht, bietet sich die Anpassung der Methode „learn“ der Klasse „FlashcardsWindow“ wie folgt an:*

```
public void learn() {
          Object message = JOptionPane.showInputDialog(
                              frame,
                              "Please choose the learning strategy:",
                              "",
                              JOptionPane.INFORMATION_MESSAGE,
                              null,
                              /* Object[] selectionValues*/,
                              /*Object initialSelectionValue*/);
          if (message != null) {
               /* message contains selected learning strategy*/
          }
}
```