Proseminar Programmiermethodik
Universität Innsbruck - Institut für Informatik
B. Murauer, D. Silbernagl, M. Tschuggnall

DBIS
DATABASES AND INFORMATION SYSTEMS

20.06.2017

Übungsblatt 10

Aufgabe 1 (Design Patterns: Abstract Factory – 3 Punkte)

Stellen Sie sich vor, Sie sind Mitarbeiter in einem Shop und wollen in Ihrem System Geräte verwalten bzw. erzeugen können. Da der Laden noch in seinen Anfängen steckt, kann im System nur unterschieden werden, ob ein Gerät ein Laptop oder ein Telefon ist. D.h. weitere Produkte kennt es noch nicht.
Für dieses Übungsbeispiel benötigen wir ausschließlich die Namen der Geräte. Wenn das z.B. vom Kunden gewünschte Gerät ein Laptop ist, kennt das System derzeit nur die zwei Marken "Lenovo" und "HP". Falls das Gerät ein Telefon ist, soll das System lediglich zwischen dem Hersteller "Apple" und "Google" unterscheiden.
Verwenden Sie nun zur Umsetzung dieser Sachverhalte das **Abstract Factory** Design Pattern. Die *Abstract Factory* soll entweder eine *LaptopFactory* oder eine *PhoneFactory* mittels Verwendung der *DeviceFactory* und dem darin übergebenen Typ. Zur Unterscheidung des Typ, also der Kategorien (Laptop oder Telefon) können Sie Strings verwenden. Die *LaptopFactory* und *PhoneFactory* sind Subklassen von der *DeviceFactory*.

Implementieren Sie alle notwendigen Klassen und erstellen Sie zum Schluss eine Testklasse, in der Sie Ihre Implementation des Design Patterns demonstrieren. Erklären Sie in einer Textdatei kurz die Eigenschaften bzw. Vor- und Nachteile dieses Design Patterns und finden Sie ein weiteres Anwendungsbeispiel dafür.

Aufgabe 2 (Design patterns: Composite Pattern – 2 Punkte)

Zur Dateisystemverwaltung auf einem Rechner wird meist eine Ordnerstruktur verwendet. Erstellen Sie eine Klasse **Folder**, in der eine Liste von Ordnern gespeichert werden kann. Fügen Sie sinnvolle Attribute, wie z.B. einen Namen hinzu.
Verwenden Sie nun zur Umsetzung dieses Sachverhalts das **Composite** Design Pattern. Erstellen Sie eine Testklasse, in der Sie Ihre Implementation des Design Patterns demonstrieren. Geben Sie am Ende den Pfad der Ordnerstrukturen aus, wie beispielsweise */home/user/bin*. Erklären Sie in einer Textdatei kurz die Eigenschaften bzw. Vor- und Nachteile dieses Design Patterns und finden Sie ein weiteres Anwendungsbeispiel dafür.

Aufgabe 3 (Design patterns: Observer Pattern – 5 Punkte)

Sehen Sie sich die Implementierung der Klasse `Game` (OLAT) an, die das Beispiel aus dem letzten Übungszettel enthält. Darin ist das Schere-Stein-Papier-Spiel (mit Erweiterung) implementiert. Erweitern Sie nun den Code, um eine Statistik zum aktuellen Spielverlauf erzeugen zu können. Überlegen Sie hierbei, wie das Observer-Pattern in diesem Code verwendet werden könnte. Ändern Sie den Code so ab, dass das Pattern verwendet wird. Erklären Sie in einer Textdatei kurz die Eigenschaften bzw. Vor-

und Nachteile dieses Design Patterns und finden Sie ein weiteres Anwendungsbeispiel dafür. Die Statistik soll nach jeder Runde den aktuellen Zwischenstand in folgender Form ausgeben:

```
*****Statistics*****
Rounds played: 6
Player1-#wins: 1
Player2-#wins: 3
#Draws: 2
********************
```

Wichtig: Laden Sie bitte Ihre Lösung (.txt, .java oder .pdf) in OLAT hoch und geben Sie mittels der Ankreuzliste auch unbedingt an, welche Aufgaben Sie gelöst haben. Die Deadline dafür läuft am Vortag des Proseminars um 23:59 (Mitternacht) ab.