Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. M. Hölzl, J. Rinke

WS 2015/16
Übungsblatt 9

Übungen zu Softwaretechnik: Programmierung und Software-Entwicklung

Aufgabe 9-1 Exkursion zu Java Reflections API *Präsenz*

a) Wann ist die Anwendung von Reflections (in Java) generell sinnvoll? Fallen Ihnen konkrete Anwendungsfälle ein?

b) Reflections sind sehr mächtig und daher auch in Java mit Vorsicht zu genießen. Welche kritischen Punkte fallen Ihnen ein?

c) In Java erzeugt die JVM für jedes Objekt eine Instanz der Klasse java.lang.Class, welche Methoden bereitstellt, die es ermöglichen die Eigentschaften eines Objekts zur Laufzeit zu untersuchen. Class ist der Eintrittspunkt für alle Reflection APIs in Java. Für jedes instantiierte Objekt kann man mit `Object.getClass()` die Klasse des Objekts bekommen. Implementieren Sie eine Klasse `Foo`, erzeugen Sie ein Objekt dieser Klasse, rufen Sie die `getClass()`-Methode auf und erzeugen Sie eine Ausgabe. Geben sie außerdem folgende Zeile aus:
`System.out.println("foo".getClass());`
Wie lauten die beiden Ausgaben?

d) Schreiben Sie außerdem eine `print(Object)`-Methode, die den Namen des übergebenen Objekts auf der Konsole ausgibt.

e) Außerdem kann man Informationen über die einzelnen Member (Field, Method, Constructor) einer Klasse bekommen und diese verändern. Fügen Sie in die Klasse Foo ein (private) String-Field „name“ ein und schreiben Sie in ihrer Klasse Foo eine `print()`-Methode, die den Inhalt des Fields auf der Konsole ausgibt. Ändern Sie nun die Ausgabe indem Sie mittels Reflection den Inhalt des Fields ändern. Was müssen Sie alles ändern um die Ausgabe der Methode zu verändern?

f) Wie Sie in Teilaufgabe e) gesehen haben, kann man durch Java-Reflection auf den Inhalt eines Fields zugreifen. Wie bereits erwähnt kann man aber auch Methoden aufrufen. Um das zu machen kann man zum Beispiel mittels der `getMethod(name, params ...)`-Methode auf die Methode „name“ einer Klasse (die diese Methode hoffentlich implementiert) zugreifen. Rufen sie die `print()`-Methode mit Reflection auf.

Aufgabe 9-2 **Creational Patterns** *Präsenz*

a) Mittlerweile haben Sie fünf Creational Patterns kennen gelernt. Wie lauten die Creational Patterns und beschreiben Sie knapp die Ziele, die unter Verwendung des jeweiligen Patterns erreicht werden sollen.

b) Was könnten im Zusammenhang mit der Instantiierung von Objekten so genannte **Factories** (auch Simple Factory oder Factory Pattern genannt) sein? Als Hilfestellung soll Ihnen Abbildung 1, die es an den entsprechenden Stellen zu vervollständigen gilt, dienen.

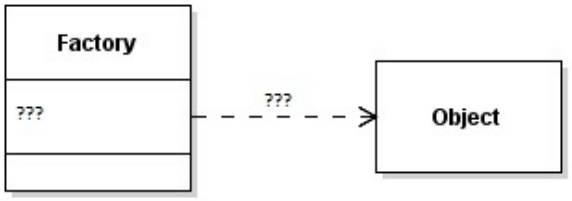

Abbildung 1: Ergänzen Sie die Abbildung sinnvoll

c) Geben Sie ein möglichst einfaches Beipsiel für eine Factory an, die ein Objekt der Klasse Object erzeugt.

d) Welcher Vorteil für die Instantiierung von Objekten einer Klasse an unterschiedlichen Stellen im Code fällt Ihnen auf, wenn sie annehmen, dass für die Instantiierung eine Factory verwendet wird? Betrachten Sie insbesondere den Fall, dass zum Beispiel eine andere Subklasse instantiiert werden soll.

e) Welches Ihnen bekannte Pattern könnte auch als Beispiel für die Umsetzung des Factory-Ansatzes dienen?

f) Sind Factories immer ein Beispiel für die Umsetzung des „Factory Method“-Pattern (nach Gang of Four)?

g) Geben Sie basierend auf dem nun gesammelten Vorwissen und der Beschreibung aus dem Buch „Design Patterns“ ein UML-Klassendiagramm für das Design Pattern „Factory Method“ an.
„Define an interface for creating a certain type of object; let subclasses decide which concrete class should be instantiated.“
Fällt Ihnen eine Ähnlichkeit zu einem bekannten Behavioral Pattern auf und wenn ja zu welchem?

h) Jetzt kenne sie das „Factory Method“-Pattern und haben zumindest bereits eine grobe Idee vom „Abstract Factory“-Pattern. Was ist der Unterschied zwischen den beiden Design Pattern?

i) Oftmals wird im Laufe der Entwicklung aus der Anwendung des „Factory Method“-Pattern eine Umsetzung des „Abstract Factory“-Pattern. Um sich den Unterschied nochmal zu veranschaulichen geben Sie ein einfaches Beispiel für das „Factory Method“-Pattern und ein einfaches Beispiel für das „Abstract Factory“-Pattern an.