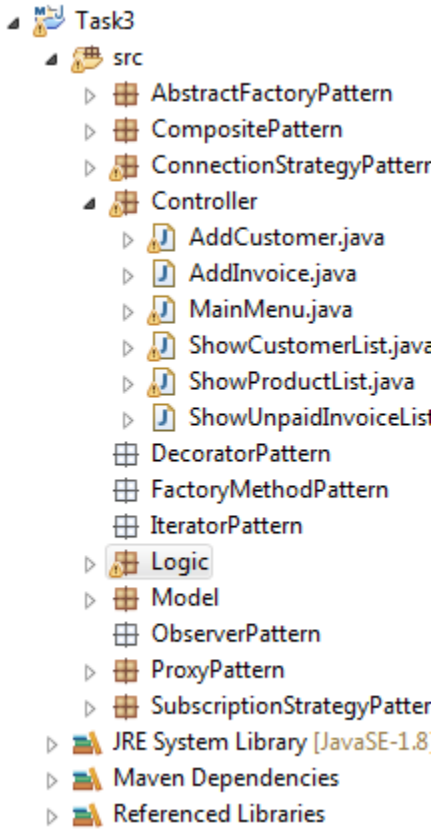
VU Software Engineering 2
VU-Leiter: Prof. Uwe Zdun
Gruppenarbeit Task 3 – Design Patterns in Practice 2

TASK 3 – KEY DESIGN PRINCIPLES

Team 1, Gruppe 3

Cordula Eggerth (0750881), a0750881@unet.univie.ac.at
Sandra Hofmarcher (1404086), a01404086@unet.univie.ac.at
Jasmin Klementschitz (1328827), a01328827@unet.univie.ac.at
Martin Regenfelder (1104500), a01104500@unet.univie.ac.at

Hinsichtlich der Key Design Principles wird zunächst die Modularisierung des Projekts angestrebt, indem mehrere Packages, wie z.B. AbstractFactoryPattern, CompositePattern, Logic, Controller erstellt werden.
Hier ein Ausschnitt aus der Struktur des Projekts:



Die jeweiligen Packages haben einen Name, der leicht auf ihre Funktionalität schließen lässt, und sie enthalten z.B. im Falle des Package Controller alle Klassen, die Benutzereingaben handhaben und vom GUI entgegennehmen und

für die Verarbeitung weitergeben. Es werden dort auch fehlerhafte Eingaben abgefangen und entsprechend gehandhabt. Im Gegensatz dazu ist z.B. das Package ConnectionStrategyPattern damit beschäftigt, die jeweils angeforderte ConnectionFactoy zu erstellen, und das Strategy Pattern angemessen umzusetzen.

Das heißt, es erfolgt im Projekt eine klare Trennung der Aufgabenbereiche („Separation of Concerns“) der einzelnen Packages und darin eine Aufteilung der Verantwortungsbereiche zwischen den Klassen und Interfaces.

Jede Klasse und jedes Interfaces hat seinen/ihren klaren Aufgabenbereich. Dies wird auch durch die entsprechenden Javadoc Comments verdeutlicht und beschrieben, damit es für Programmierer, die neu im Projekt sind, verständlich ist. Dies wird auch anhand des untenstehenden Beispiels veranschaulicht:

```java
package ConnectionStrategyPattern;

import AbstractFactoryPattern.FastBillConnectionFactory;

/**
 * FastBillConnectionStrategy implementiert das Interface IConnectionStrategy. Es legt eine neue
 * FastBillConnection an und gibt diese an die aufrufende Klasse zurueck. Die Klasse ist Teil des
 * ConnectionStrategyPattern.
 * @author Cordula Eggerth
 *
 */
public class FastBillConnectionStrategy implements IConnectionStrategy {

    /**
     * execute legt eine neue FastBillConnection an und gibt diese zurueck.
     * @param mode
     * @return IConnection
     */
    @Override
    public IConnection execute(String mode) {

        IConnectionFactory factory = null;
        factory = new FastBillConnectionFactory();
        IConnection connection = factory.createConnection();
        return connection;

    }

}
```

Am Beispiel des Codes der Klasse FastBillConnectionStrategy wird klar, dass die Separation of Concerns erfolgt und ein gewisser Bereich des Strategy Patterns hier abgeabeitet wird. Es wird der klar abgegrenzte Aufgabenbereich der Erstellung einer FastBillConnectionFactory, über die die entsprechende Connection geliefert wird.

Hinsichtlich der Key Design Principles werden vom vorliegenden Projekt auch die Prinzipien des Information Hiding und der Kapselung erfüllt. Information Hiding bedeutet, dass der Zugriff auf die Instanzvariablen von außen über Get- und Set-Methoden erfolgt. Dies wird auch im untenstehenden Code-Beispiel aus der Klasse Customer im Package Model gezeigt:

Customer.java

```
41      /**
42       * get Id des Customer
43       * @return id
44       */
45      public int getId() {
46          return id;
47      }
48
49      /**
50       * set Id des Customer
51       * @param id
52       */
53      public void setId(int id) {
54          this.id = id;
55      }
56
57      /**
58       * get Customer Type
59       * @return customerType
60       */
61      public String getCustomerType() {
62          return customerType;
63      }
64
65      /**
66       * set Customer Type
67       * @param customerType
68       */
69      public void setCustomerType(String customerType) {
70          this.customerType = customerType;
71      }
72
```

Durch das Prinzip des Infromation Hiding sollen die Module in sich konsistent sein und sollen vor direkten Zugriffen von außen geschützt werden, was eine vorteilhafte Kapselung bedeutet. Die Instanzvariablen und internen Details sollen also nicht direkt von außen zugreifbar sein und an alle andern Module gezeigt werden.
Ein „gutes“ Projekt hat somiz eine hohe Bindung intern zwischen den Module, was heißt, dass die Kohäsion hoch ist. Extern haben „gute“ Module jedoch untereinander eine lose Koppelung (Coupling), und die Kommunikation passiert nur über entsprechend gezielt definierte Schnittstellen. Dies wird auch im vorliegenden Projekt umgesetzt.

Quelle:

Prof. Uwe Zdun. Foliensatz Key Design Principles.