Übung 11:
Design Patterns 2

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Software Engineering
WS 2018/19 - Dr. Michael Eichberg

Abgabe

Die Übung wird als *sbt*-Projekt (`http://www.scala-sbt.org`) bereitgestellt. *sbt* kann verwendet werden, um Java (und Scala) Projekte einfach auszuführen und zu testen. Installieren Sie `sbt` auf Ihrem Rechner und stellen Sie sicher, dass ein Java SDK (min. Java 8) installiert ist. Überprüfen Sie, dass der Java Compiler `javac` auf der Kommandozeile ausführbar ist, auch wenn Sie sich nicht im Verzeichnis mit der ausführbaren Datei befinden. Wenn nicht, passen Sie Ihren `PATH` entsprechend an, Hinweise wie dies bei Ihrem Betriebssystem möglich ist, finden Sie im Internet.
IntelliJ erlaubt das Importieren von *sbt*-Projekten direkt, dafür muss allerdings das Scala Plugin installiert sein. Um ein Eclipse-Projekt zu erzeugen, kann `sbt eclipse` im Projektverzeichnis (das Verzeichnis, dass die Datei build.sbt enthält) ausgeführt werden, danach kann das Projekt mit `Datei > Importieren > Vorhandene Projekte in Arbeitsbereich` importiert werden. Wenn Sie eine `main`-Methode geschrieben haben, können Sie Ihr Programm mit `sbt run` ausführen, Tests können Sie unter `src/test/java/` anlegen und mit `sbt test` ausführen.
Das Ausführen des Kommandos `sbt` im Projektverzeichnis startet den interaktiven Modus von *sbt*. Im interaktiven Modus können Kommandos ausgeführt werden, ohne jedes Mal `sbt` eingeben zu müssen.
Um Ihre Lösung abzugeben, melden Sie sich zunächst unter

`https://submission.st.informatik.tu-darmstadt.de/course/se18`

an und erzeugen Sie ein *submission token*. Wenn Sie den interaktiven Modus von *sbt* verwenden, führen Sie dann folgendes Kommando aus:

`submit <ihreTUId> <submissionToken>`

wobei *<ihreTUId>* Ihre eindeutige Identifikationsnummer an der TU Darmstadt (**nicht Ihre Matrikelnummer!**) und *<submissionToken>* das zuvor generierte Token ist. Geben Sie die spitzen Klammern nicht mit an, der Befehl sollte etwa so aussehen: `submit ab01cdef 01234567`. Wenn Sie nicht den interaktiven Modus verwenden, muss das Kommando in Anführungszeichen gesetzt werden, also `sbt "submit <ihreTUId> <submissionToken>"`. Sie können (innerhalb der Abgabefrist) beliebig oft eine Lösung einreichen, allerdings wird nur die zuletzt eingereichte Lösung bewertet. Die letzte Lösung, die ein Gruppenmitglied eingereicht hat, wird zur Bewertung für die ganze Gruppe herangezogen. Koordinieren Sie sich daher in Ihrer Gruppe, wer Ihre gemeinsame Lösung einreicht.
Stellen Sie sicher, dass Sie das zur Verfügung gestellte Template nutzen und die Namen und Signaturen der vorgegeben Klassen und Methoden nicht verändern sowie dass von Ihnen hinzugefügte Klassen und Methoden die geforderten Namen und Signaturen verwenden. Ändern Sie außerdem nichts an der vorgegebenen Datei `build.sbt`. Andernfalls wird das System Ihre Lösung nicht bewerten können. Beachten Sie, dass der Zugriff auf die Abgabeplattform nur im **internen Netz der Universität** möglich ist. Für einen Zugriff von außerhalb benötigen Sie daher eine VPN-Verbindung. Überprüfen Sie, ob Ihre Abgabe erfolgreich war, indem Sie sie von der Abgabeplattform herunterladen. Überprüfen Sie dabei auch, ob alle Dateien in der Abgabe enthalten sind.

Einführung

In dieser Übung befassen Sie sich mit weiteren Design Patterns. Bearbeiten Sie die folgenden beiden Aufgaben und erstellen Sie eine einzelne PDF-Datei mit Ihren Antworten. Legen Sie diese Datei in den Ordner `solution` in dem Template, das Ihnen zur Verfügung gestellt wurde, und verwenden Sie zur Abgabe wie oben angegeben `sbt submit`.

Problem 1 Design Pattern erkennen — 10P

a) Im zur Verfügung gestellten Template finden Sie eine Programm zur Verwaltung von Bankkonten. Identifizieren Sie in diesem Template verwendete, Ihnen bekannte Design Pattern und nutzen Sie die Annotationen aus dem Package `ex11.annotations` um die einzelnen Komponenten der identifizierten Patterns (Klassen und Methoden) zu annotieren.

b) Erstellen Sie ein Klassendiagramm zum vorhandenen Template (nicht jedoch zu den Annotationen aus dem Package `ex11.annotations`). Machen Sie darin Interfaces und abstrakte Klassen deutlich ersichtlich. Berücksichtigen Sie alle Felder und Methoden.

Problem 2 Observer Pattern implementieren — 6P

a) Die Bank hinter dem System möchte die Funktionalität um ein **Depot** erweitern. Einem Depot können **Wertpapiere** hinzugefügt werden. Momentan möchte die Bank nur das Handeln mit **Aktien** als Wertpapiere zulassen, Ihr Design soll aber die Erweiterung um andere Wertpapiere ermöglichen. Wertpapiere benötigen einen Namen, eine ID und einen Wert, den man manipulieren kann. Die Manipulation des Wertes erfolgt über eine Methode, die einen Wert entgegennimmt und diese auf den momentanen Wert addiert. Verwenden Sie zur Modellierung von Werten den Typ `double`. Zusätzlich sollen Wertpapiere die Möglichkeit besitzen, registrierte Observer zu informieren, falls dies durch eine Änderung des Wertes erforderlich ist. Ändert sich der Wert eines sich im Depot befindlichen Wertpapiers, soll das Depot benachrichtigt werden und daraufhin seinen momentanen Gesamtwert berechnen und über die Kommandozeile ausgeben.
Erstellen Sie die neue(n) Klasse(n) im bereits existierenden Package `ex11.depot`. Sie können sich bei der Implementierung an den bereits gegebenen Klassen im Template orientieren.

b) Annotieren Sie wie in Problem 1 Klassen und Methoden mit den passenden Annotationen für das Observer Pattern.