Universität Siegen
Fachgruppe Praktische Informatik

Übung Testgetriebene Entwicklung (TDD)

Blatt 1 – Ausgabe am 13.06.2017, Bearbeitung bis **27.06.2017**

Hinweise: Verwenden Sie für die Lösung der Aufgabe das *JUnit-Framework*. Die integrierte Entwicklungsumgebung *Eclipse* bietet eine vollständige Integration des Frameworks u.a. mit entsprechenden Hilfsfunktionen zur Erstellung von Testklassen und bedarf keiner weiteren Konfiguration. Eine aktuelle Eclipse Version erhalten Sie unter https://www.eclipse.org/downloads/.
Die Abgabe erfolgt über das Repository. Erstellen Sie im Verzeichnis `tdd` ein Unterverzeichnis und benennen Sie dieses nach Ihnen selbst ([*Nachname Vorname* ohne Leerzeichen). Nutzen Sie das Repository auch dazu, Zwischenstände Ihrer Arbeit zu sichern.
Bei Fragen wenden Sie sich bitte an cpietsch@informatik.uni-siegen.de.

Aufgabe 1.1 *TDD: Einarbeitung*

Das Projekt `2016s_pep_junit_money` beinhaltet das Grundgerüst einer *digitalen Geldbörse* (*Money Bag*), welche in der Lage ist Geldbeträge verschiedener Währungen aufzunehmen und diverse Operationen auf diesen auszuführen.
Importieren Sie das Projekt in Ihre Eclipse Arbeitsumgebung und machen Sie sich mit den jeweiligen Klassen und Methoden vertraut. Eine informelle Beschreibung der zentralen Klassen finden Sie im Anhang **??**.

Aufgabe 1.2 *TDD: Implementierung*

Implementieren Sie die Methoden der Klassen `MoneyBag`, `Money` und `Currency` aus dem Paket `de.usi.inf.money.impl` nach dem 'Test-First'-Ansatz:

a) Erstellen Sie für jede Klasse eine Testklasse (JUnit Test Case) im Ordner `test`.
b) Implementieren Sie die Auf-/Abbau- und Testmethode(n) und führen Sie den Testfall aus. Berücksichtigen Sie ggf. Ausnahmebehandlungen.
c) Implementieren Sie die Methode(n) der jeweiligen Klasse und führen Sie den/die Tests erneut aus.
d) Erstellen Sie abschließend eine JUnit Test Suite, um alle Testfälle auf Knopfdruck auszuführen.

Nachdem alle Tests erfolgreich durchlaufen wurden führen Sie abschließend einen Integrationstest durch, indem Sie die Klasse *RunMoneyBagUI* ausführen. Das Ergebnis sollte dem aus Abbildung **??** gleichen.

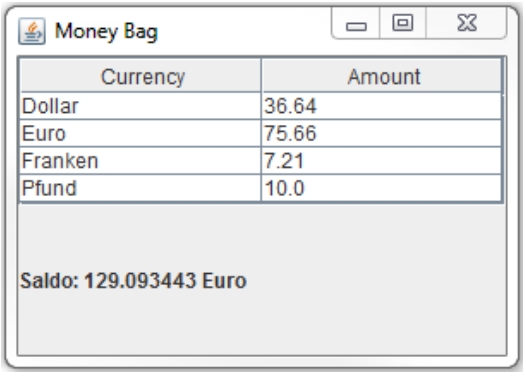

Abbildung 1: MoneyBag UI

A Anhang

Currency : Eine Währung (`Currency`) besitzt einen Namen (`#name`) und eine Menge von Wechselkursen (`#rates`) in Form einer Map. Der Schlüsselwert entspricht dabei der Bezeichnung einer anderen Währung und der Wert dem jeweiligen Wechselkurs. Beim Anlegen einer neuen Währung sollen im späteren Verlauf die aktuellen Wechselkurse aus einer Datenbank ausgelesen werden. Da die Datenbank zum aktuellen Zeitpunkt noch nicht existiert wurde die *externe Abhängigkeit* durch einen *Stub* ersetzt. Dazu wurde eine *Indirektionsschicht* in Form einer Fabrikklasse (`ExchangeRatesDataBaseAdapterFactory`) und einem entsprechenden Interface (`IExchangeRatesDataBaseAdapter`) eingeführt.

- **ExchangeRatesDataBaseAdapterFactory**: Die Klasse ist für das Erzeugen und Verwalten eines Adapter vom Typ `IExchangeRatesDataBaseAdapter` verantwortlich. Die Methode `create()` erzeugt sofern das Klassenattribut `#adapter` gleich Null ist einen neuen Adapter. Ansonsten wird der existierende zurück gegeben. An dieser Stelle kann die *Setter-Methode* verwendet werden, um der Fabrikklasse einen benutzerdefinierten Adapter, in diesem Fall den Stub, zu übergeben.
- **IExchangeRatesDataBaseAdapter**: Das Interface definiert Methoden zum Verbinden und Trennen einer Verbindung mit einer Datenbank, sowie der Zugriffsmethode `select()`, welche zu einer übergebenen Währungsbezeichnung die in der Datenbank gespeicherten Wechselkurse liefert.

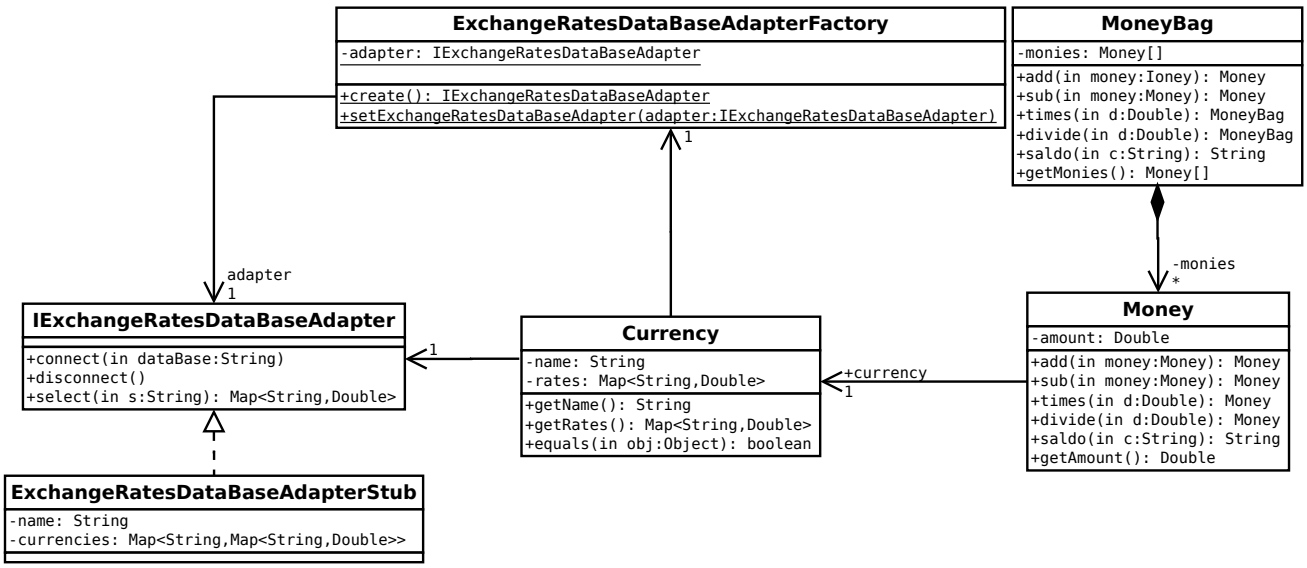

Abbildung 2: MoneyBag Klassendiagramm

Zuvor muss jedoch über die Methode `connect()` eine Verbindung aufgebaut werden.
Bemerkung: Übergeben Sie der Methode `connect()` die Zeichenkette "*ExchangeRates.txt*". Der bereitgestellte Stub unterstützt die Währungen *Dollar*, *Euro*, *Franken* und *Pfund*.

Der Konstruktor der Klasse `Currency` ruft die statische Methode `create()` der Fabrikklasse und die Methode `select()` des zurückgegebenen Adapters auf, um die Map `#rates` zu initialisieren.

Money : Ein `Money`-Objekt besitzt neben einer Währung (`#currency`) einen Betrag (`#amount`).
Des Weiteren können zu einem bestehenden Betrag weitere Beträge addiert und subtrahiert werden. Dazu muss die Währung des übergebenen Money-Objekts der des Aufrufers entsprechen. Ebenfalls möglich ist die Multiplikation und Division des aktuellen Betrags mit bzw. durch einen Multiplikanden bzw. Dividenden.
Sofern die Währung geändert wird (`setCurrency()`) muss auch der Betrag (`#amount`) angepasst werden.
Die Funktion `saldo()` berechnet anhand des übergebenen Währungsbezeichners den Betrag und gibt diesen in Form einer Zeichenkette zurück.

MoneyBag : Eine Geldbörse verwaltet mehrere Geldbeträge unterschiedlicher Währungen (`#monies`).
Des Weiteren können neue Geldbeträge hinzugefügt (`add()`) und entfernt (`sub()`) werden. Sofern die Geldbörse noch keinen Eintrag der entsprechenden Währung besitzt wird das entsprechende Money-Objekt hinzugefügt. Andernfalls wird der Betrag des übergebenen Money-Objekts zum bestehenden addiert bzw. subtrahiert.

Die Methode `times()` und `divide()` ruft die entsprechende Methode der verwalteten Money-Objekte auf.
Die Methode `saldo()` ruft ebenfalls die entsprechende Methode der verwalteten Money-Objekte auf und gibt deren Summe in der entsprechenden Währung als Zeichenkette zurück.