ÜBUNGSZETTEL MIKROARCHITEKTUR

BLUEJ-BUCH KLASSENENTWURF

Arbeite das Buchkapitel „Klassenentwurf“ (K8). Die Folien dazu sind online. Erledige alle Übungen aus dem Kapitel. Dokumentiere deine Erledigungen.

BLUEJ-BUCH ENTWURF VON ANWENDUNGEN

Erledige das BlueJ-Buch-Kapitel „Entwurf von Anwendungen“ (K15). Die Folien dazu sind online. Erledige alle Übungen aus dem Kapitel. Dokumentiere deine Erledigungen.

ENTWURFSMUSTER 1

Arbeite dich in das Thema Entwurfsmuster ein. Du sollst wissen, was Entwurfsmuster sind, wie sie beschrieben werden und wozu man sie benötigt.

Du bekommst dazu auch eine Einführung im Unterricht anhand des Strategy-Entwurfsmusters.

ENTWURFSMUSTER 2

Arbeite dich in folgende Entwurfsmuster ein:

- Chain of Responsibility
- TemplateHook
- Decorator
- Builder
- Adapter
- Observer

Einarbeiten heißt:

- Muster lernen
- Muster anhand einer minimalen Aufgabenstellung selbst implementieren
- Alles dokumentieren.

AUSGANGSPUNKT FÜR DIE FOLGENDEN AUFGABEN

Die im Laufe der folgenden Aufgaben zu erstellende Software bildet einen einfachen Währungsrechner nach, der unter Anwendung verschiedener Entwurfsmuster implementiert werden muss. Die abstrakte Klasse WR sowie die Interfaces IUmrechnen und ISammelumrechnung sind als Ausgangspunkt für die Arbeit sind im Anhang gegeben.

```
public interface IUmrechnen{
    public double umrechnen(String variante, double betrag);
}
public interface ISammelumrechnung{
     public double sammelumrechnen(double[] betraege, String variante);
}
public abstract class WR implements IUmrechnen{
    public double umrechnen(String variante, double betrag){
```

```
        ;       }
}
```

Hinweis: Wenn du die folgenden Entwurfsmuster anhand einer zur Währungsrechner-Thematik alternativen Domäne implementieren möchtest, dann ist das natürlich in Absprache mit der LV-Leitung möglich.

CHAIN OF RESPONSIBILITY

Erstellen Sie eine Verantwortlichkeitskette mit mindestens zwei Währungsrechnern (z.B. EUR2YEN, EURO2Dollar). Neben dem Standardverhalten einer Chain of Responsibility soll es möglich sein, neue Währungsrechner in die Kette aufzunehmen bzw. bestehende aus der Kette zu löschen (jeweils am Ende der Kette).

TEMPLATE HOOK

Vermeiden Sie Codeduplikate in der Methode umrechnen() in den resultierenden Unterklassen. Platzieren Sie den Code dazu soweit wie möglich und sinnvoll in der abstrakten Klasse WR und delegieren Sie ausschließlich das eigentliche umrechnen per Template-Hook-Muster an die Unterklassen.

DECORATOR

Dekorieren Sie ihre Währungsrechner-Kette mit folgenden Dekoratoren:

a) Belegung eines Umrechnungsvorganges mit Gebühren (z.B. 0,5 % des Umrechnungsbetrages)
b) Belegung eines Umrechnungsvorganges für Umrechnungen von Euro nach Währung X (nicht in die andere Richtung) mit fixen Gebühren von 5 Euro.

BUILDER

Implementieren Sie das Builder-Pattern (https://dzone.com/articles/design-patterns-the-builder-pattern) für eine Objektinstanziierung einer auf Wunsch auch dekorierten Währungsrechner-Kette.

ADAPTER

Externe Anwendungen benötigen eine Implementierung der Schnittstelle ISammelumrechnung (siehe Angabe), um Sammelumrechnungen durchführen zu können. Stellen Sie einen Adapter bereit, der Sammelumrechnungen in der geforderten Form (siehe Methodensignatur) zur Verfügung stellt und dazu die Funktionalität der (dekorierten) Währungsrechner-Kette verwendet.

OBSERVER

Am Währungsrechner sollen sich mehrere Observer registrieren können. Jedes Mal wenn eine Umrechnung stattfindet sollen alle Observer benachrichtigt werden. Alle Informationen der Umrechnungen (Ausgangsbetrag, Ausgangswährung, Zielwährung, Zielbetrag) sollen mit der Benachrichtigung versendet werden. Beispielhaft sollen zwei Observer implementiert werden:

1) Atom-Feed-Observer: Erzeugt einen Atom-Feed mit allen Umrechnungsinformationen und Zeitstempel (verwende dazu: https://mvnrepository.com/artifact/rome/rome)
2) Log-Observer: Erzeugt eine Log-Text-Datei mit allen Umrechnungsinformationen und Zeitstempel.

SOLID UND SONSTIGE DESIGNPRINZIPIEN

Lerne im Detail die die **SOLID-Prinzipien**:

https://dev.to/ham8821/solid-principles-to-start-with-object-oriented-programming-1e49

http://openbook.rheinwerk-verlag.de/oo/oo_03_prinzipien_000.htm#Xxx999144

https://en.wikipedia.org/wiki/SOLID

https://www.youtube.com/watch?v=TMuno5RZNeE

https://www.youtube.com/playlist?list=PL4CE9F710017EA77A

Lerne mindestens 3 **weitere Prinzipien** des objektorientierten Entwurfs kennen (DRY, YAGNI, KISS etc. siehe Moodle-Kurs https://moodle.tsn.at/course/view.php?id=24763#section-10).

Wenn du alles im Detail gelernt hast, **weise an verschiedenen Stellen deiner Implementierungen** aus den vorhergehenden Aufgaben **die Realisierung oder die Verletzung** der SOLID-Prinzipien bzw. der übrigen Designprinzipien (DRY, YAGNI, KISS etc. siehe Moodle-Kurs) nach.