

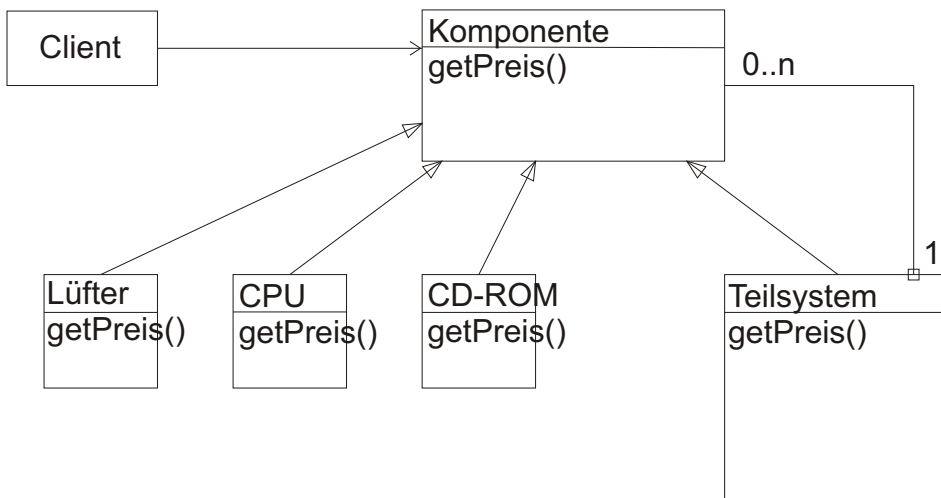
Blatt 6

Aufgabe 10: Kompositum-Pattern

Entwickeln Sie ein Klassenmodell, mit dem man die Struktur eines Rechnersystems abbilden kann.

- Ein Rechnersystem besteht aus einer einzelnen Komponente (mit Namen und Preis) oder aus einem Teilsystem
- Ein Teilsystem besteht aus einzelnen Komponenten und kann wiederum Teilsysteme besitzen.
- Sowohl für einzelne Komponenten wie auch für Teilsysteme kann ein Preis angegeben werden.

a) Erstellen Sie ein Klassenmodell mit seinen erforderlichen Beziehungen



b) Spezifizieren Sie die erforderlichen Methoden, um den Preis des Gesamtsystems zu bestimmen und ein Hardwaresystem zu konstruieren

c) Implementieren Sie das Klassendiagramm in Java und schreiben Sie ein Hauptprogramm, das Ihre Klassen testet.

```
public class Komponente
{
    String bezeichnung;
    double preis;
    Komponente Teilsystem = null;

    // Constructor
    public Komponente
    {
    }

    public double getPreis
    {
        double gesamt = preis;

        if (Teilsystem == null)
            gesamt += Teilsystem.getpreis();
        return preis;
    }
}
```

Aufgabe 11

Bei manchen Klassen ist es wichtig, dass es von diesen nur **ein** Objekt (Exemplar) geben darf. Oft sind dies Klassen, die zentrale Systemdienste zur Verfügung stellen, z.B.

- Container-Klassen zur Verwaltung von Objekten soll es nur einmal geben
- Eine Klasse, die bei einem Brettspiel die Steuerung des Spielablaufs übernimmt (oder die Klasse, die die Spielsituation verwaltet).

Würde man im Laufzeitsystem mehrere Instanzen erzeugen, so hätte man große Probleme, deren Konsistenz sicherzustellen.

Überlegen Sie sich ein Entwurfsmuster für eine Klasse **Singleton**, die nur **ein einziges** Objekt besitzen darf.

- Stellen Sie sicher, dass nur ein einziges Objekt dieser Klasse erzeugt werden kann.
- Realisieren Sie einen zentralen Zugriffspunkt, d.h. über eine Methode soll entweder das Objekt erzeugt werden (falls es noch keins gibt) oder eine Referenz auf das bereits erzeugte Objekt geliefert werden.

a) Implementieren Sie Ihre Lösung in Java.

```
public class Singleton
{
    private static Singleton myInstance = null;
    private Object mData;
    private Singleton() { }

    public static Singleton getInstance()
    {
        if (myInstance == null)
            myInstance = new Singleton();
        return myInstance;
    }
}
```

b) Zeichnen Sie ein entsprechendes UML-Modell.