Musterlösung und Erläuterungen S. 216|2

Aufgabentext:

2 Menüwahl

Ein Menü besteht aus Vor-, Haupt und Nachspeise.

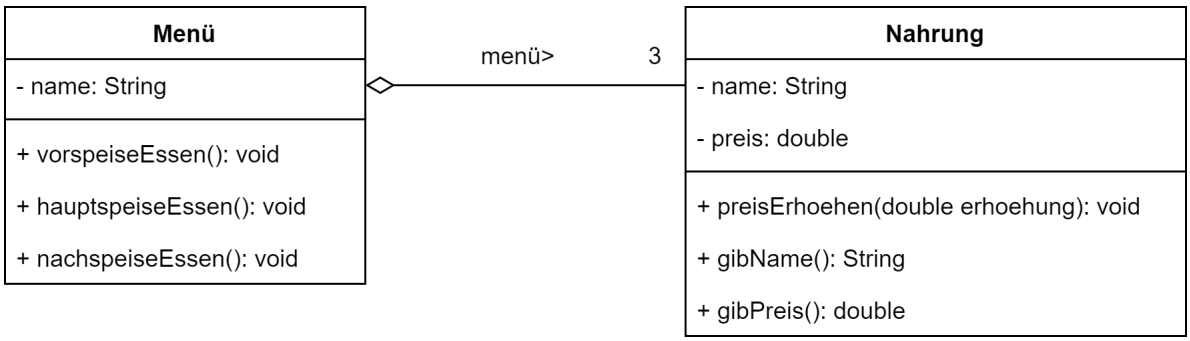
a) Gib das zugehörige Klassendiagramm an, wenn es sich bei den drei Gängen jeweils um Objekte der gleichen Klasse NAHRUNG handelt bzw. wenn zwischen den Klassen VORSPEISE, HAUPTSPEISE und NACHSPEISE unterschieden wird.

b) Implementiere für die Klassen NAHRUNG, VORSPEISE, HAUPTSPEISE und NACHSPEISE jeweils die Klassendefinition.

c) Zwei Kollegen gehen gemeinsam in der Kantine essen. Illustriere anhand von Objektdiagrammen, unter welchen Umständen sie das gleiche bzw. dasselbe Menü zu sich nehmen.

d) Ergänze in deiner Implementierung die Klasse `Menue` sowie eine Klasse `Kantine`, welche die beiden Fälle aus Teilaufgabe c) durch Variation der Parameterwerte des Konstruktors der Klasse `Menue` realisiert.

a) und b) In jedem der angesprochenen Fälle brauchen wir eine Klasse Menü, es ändern sich lediglich die Referenzen auf die entsprechenden Vorspeisen, Hauptgerichte bzw. Nachspeisen. Die zusätzlichen Attribute bzw. Methoden sind nicht von der Aufgabe vorgegeben, dienen aber der weiteren Veranschaulichung der Situation.
Achtung: Die Attribute, die Referenzen auf Objekte anderer Klassen darstellen (im ersten Fall z.B. auf Objekte der Klasse Nahrung), werden üblicherweise nicht mehr explizit im Klassendiagramm angegeben (anders als im Buch auf S. 209 z.B.), da diese Information bereits in der Verbindungslinie steckt (eine zusätzliche Angabe ist in diesem Sinne aber nicht falsch, nur unnötig, da doppelte Information abgebildet wird). **Erster Fall:**



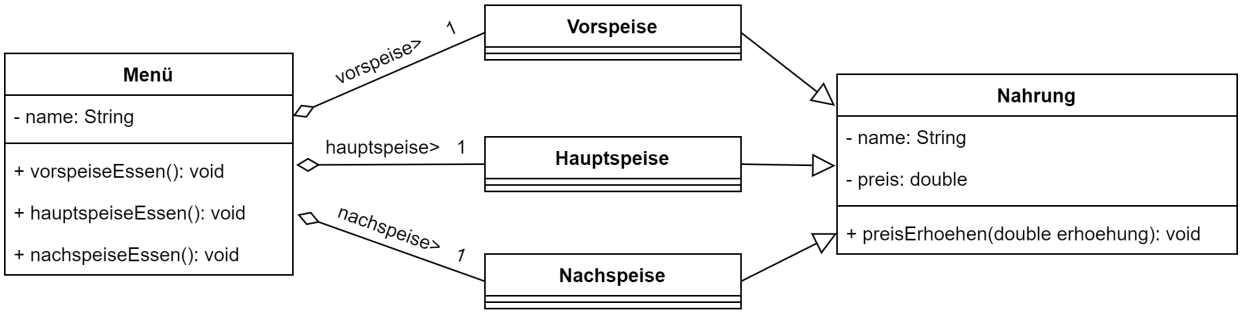
Wenn man sich das Klassendiagramm ansieht - und die Aufgabe vorher nicht liest - würde man schlussfolgern, dass ein Menü drei Objekte vom Typ Nahrung enthält. Es wird dabei keine Vorgabe gemacht, wie diese Referenzen auf die Nahrungsobjekte gespeichert werden! Man könnte hier z.B. implementieren:

```
public class Menü {
    private Nahrung vorspeise;
    private Nahrung hauptgericht;
    private Nahrung nachspeise;

    public Menü() {
        vorspeise = new Nahrung(...);
        hauptgericht = new Nahrung(...);
        nachspeise = new Nahrung(...);
    }
}
```

Der entsprechende Konstruktor für Nahrung muss natürlich noch definiert werden. Alternativ könnte man die drei Teile des Menüs aber auch in einem Feld speichern:

```java
public class Menü {
    private Nahrung[] menü;

    public Menü() {
        menü = new Nahrung[3];
        menü[0] = new Nahrung(...) // Vorspeise
        menü[1] = new Nahrung(...) // Hauptspeise
        menü[2] = new Nahrung(...) // Nachspeise
    }
}
```

Die zweite verlangte Variante legt die Verwendung einer Oberklasse nahe, da Methoden wie preisErhöhen() oder auch gibName() natürlich für alle Speisen sinnvoll sind (für dieses minimale Beispiel ist natürlich fraglich, ob wir wirklich Unterklassen brauchen, in der „Realität“ sicher nicht!), deswegen ergibt sich folgendes Klassendiagramm:



Für diese Variante sieht die Implementierung wie folgt aus (bereits mit den Zusatzmethoden für Nahrung):

```java
public class Nahrung {

    private String name;
    private double preis;

    public Nahrung(String name, double preis) {
        this.name = name;
        this.preis = preis;
    }

    public String gibName() {
        return name;
    }

    public double gibPreis() {
        return preis;
    }

    public void preiserhoehen(double erhoehung) {
        preis += erhoehung;
    }
}
public class Vorspeise extends Nahrung {

    public Vorspeise(String name, double preis) {
        super(name, preis);
    }
}
public class Hauptspeise extends Nahrung {
```

```
    public Hauptspeise(String name, double preis) {
        super(name, preis);
    }
}
public class Nachspeise extends Nahrung {
    public Nachspeise(String name, double preis) {
        super(name, preis);
    }
}
```

c) Die Aufgabe spielt darauf an, dass zwei Menüs auf dasselbe oder auf unterschiedliche Objekte zeigen können, die gesuchte Veranschaulichung sieht so aus:

Die beiden Objekte **menü1** und **menü2** zeigen beide auf jeweils dasselbe Objekt der Klasse Vorspeise, Nachspeise und Hauptspeise. Das **menü3** zeigt zwar auf gleichartige Objekte, aber nicht auf dieselben!

d) Die ergänzten Klassen sehen z.B. so aus:

```
    public class Menü {
    private String name;
    private Vorspeise vorspeise;
    private Hauptspeise hauptspeise;
    private Nachspeise nachspeise;

    public Menü(String name, Vorspeise vorspeise, Hauptspeise hauptspeise,
     Nachspeise nachspeise) {
        this.vorspeise = vorspeise;
        this.hauptspeise = hauptspeise;
        this.nachspeise = nachspeise;
        this.name = name;
    }

    public void vorspeiseEssen() {
        if(vorspeise != null) {
            System.out.println("lecker -  " + vorspeise.gibName());
            vorspeise = null;
        } else {
            System.out.println("Die Vorspeise wurde schon gegessen!");
        }
    }

    public void hauptspeiseEssen() {
        if(vorspeise != null) {
            System.out.println("lecker - " + hauptspeise.gibName());
            hauptspeise = null;
        } else {
            System.out.println("Die Hauptspeise wurde schon gegessen!");
```

```
        }
    }

    public void nachspeiseEssen() {
        if(vorspeise != null) {
            System.out.println("lecker - " + nachspeise.gibName());
            nachspeise = null;
        } else {
            System.out.println("Die Nachspeise wurde schon gegessen!");
        }
    }
}

public class Kantine {

    public Kantine() {
        Hauptspeise hauptspeise1 = new Hauptspeise("Schnitzel", 9.6);
        Hauptspeise hauptspeise2 = new Hauptspeise("Schnitzel", 9.6);
        Vorspeise vorspeise1 = new Vorspeise("Suppe", 5.2);
        Vorspeise vorspeise2 = new Vorspeise("Suppe", 5.2);
        Nachspeise nachspeise1 = new Nachspeise("Apfelstrudel", 6.10);
        Nachspeise nachspeise2 = new Nachspeise("Apfelstrudel", 6.10);
        Menü menü1 = new Menü("Super Menü", vorspeise1, hauptspeise1, nachspeise1);
        Menü menü2 = menü1;
        Menü menü3 = new Menü("Super Menü", vorspeise2, hauptspeise2, nachspeise2);
    }
}
```

Die Essens-Methoden sind dabei natürlich nicht verlangt gewesen, waren aber im obigen Klassendiagramm erwähnt worden, deswegen ist eine Beispielimplementierung dabei.