Einzelprüfung „Datenbanksysteme / Softwaretechnologie (vertieft)“

Einzelprüfungsnummer 66116 / 2015 / Frühjahr

Thema 1 / Teilaufgabe 2 / Aufgabe 3 (*Kunden und Angestellte einer Firma*)

Stichwörter: Implementierung in Java, Vererbung, Interface, Abstrakte Klasse, Adapter

---

In dieser Aufgabe implementieren Sie ein konzeptionelles Datenmodell für eine Firma, die Personendaten von Angestellten und Kunden verwalten möchte. Gegeben seien dazu folgende Aussagen:

- Eine *Person* hat einen *Namen* und ein *Geschlecht* (männlich oder weiblich).
- Ein *Angestellter* ist eine *Person*, zu der zusätzlich das monatliche *Gehalt* gespeichert wird.
- Ein *Kunde* ist eine *Person*, zu der zusätzlich eine *Kundennummer* hinterlegt wird.

(a) Geben Sie in einer objektorientierten Programmiersprache Ihrer Wahl (geben Sie diese an) eine Implementierung des aus den obigen Aussagen resultierenden konzeptionellen Datenmodells in Form von **Klassen** und **Interfaces** an. Gehen Sie dabei wie folgt vor:

- Schreiben Sie ein Interface `Person` sowie zwei davon erbende Interfaces `Angestellter` und `Kunde`. Die Interfaces sollen jeweils lesende Zugriffsmethoden (Getter) die entsprechenden Attribute (Name, Geschlecht, Gehalt, Kundennummer) deklarieren.

Lösungsvorschlag

```java
public interface Person {
  String getName();

  char getGeschlecht();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/Person.java`

```java
public interface Angestellter extends Person {
  int getGehalt();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/Angestellter.java`

```java
public interface Kunde extends Person {
  int getKundennummer();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/Kunde.java`

- Schreiben Sie eine abstrakte Klasse `PersonImpl`, die das Interface `Person` implementiert. Für jedes Attribut soll ein Objektfeld angelegt werden. Außerdem soll ein Konstruktor definiert werden, der alle Objektfelder initialisiert.

Lösungsvorschlag

```java
public abstract class PersonImpl implements Person {
  protected String name;
  protected char geschlecht;

  public PersonImpl(String name, char geschlecht) {
    this.name = name;
    this.geschlecht = geschlecht;
  }

  public String getName() {
    return name;
  }

  public char getGeschlecht() {
    return geschlecht;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/PersonImpl.java`

- Schreiben Sie zwei Klassen `AngestellterImpl` und `KundeImpl`, die von `PersonImpl` erben und die jeweils dazugehörigen Interfaces implementieren. Es sollen wiederum Konstruktoren definiert werden, die alle Objektfelder initialisieren und dabei auf den Konstruktor der Basisklasse `PersonImpl` Bezug nehmen.

Lösungsvorschlag

```java
public class AngestellterImpl extends PersonImpl implements Angestellter {
  protected int gehalt;

  public AngestellterImpl(String name, char geschlecht, int gehalt) {
    super(name, geschlecht);
    this.gehalt = gehalt;
  }

  public int getGehalt() {
    return gehalt;
  }

}
```

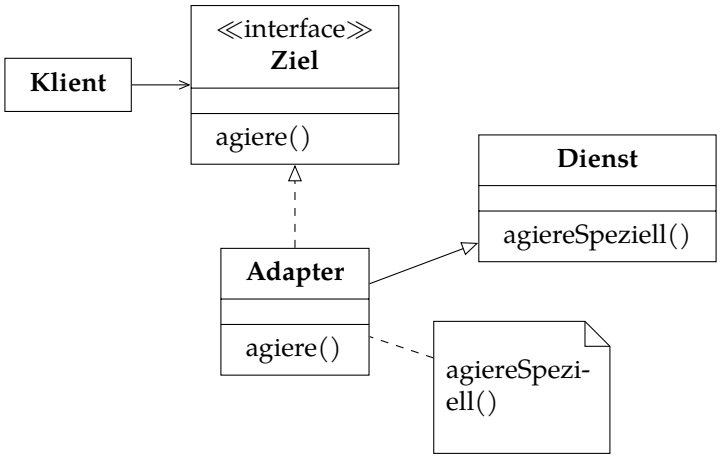
Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/AngestellterImpl.java`

```java
public class KundeImpl extends PersonImpl implements Kunde {
  protected int kundennummer;

  public KundeImpl(String name, char geschlecht, int kundennummer) {
    super(name, geschlecht);
    this.kundennummer = kundennummer;
  }

  public int getKundennummer() {
    return kundennummer;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/KundeImpl.java`

(b) Verwenden Sie das Entwurfsmuster **Adapter**, um zu ermöglichen, dass vorhandene Angestellte die Rolle eines Kunden einnehmen können. Der Adapter soll eine zusätzliche Klasse sein, die das Kunden-Interface implementiert. Wenn möglich, sollen Methodenaufrufe an den adaptierten Angestellten delegiert werden. Möglicherweise müssen Sie neue Objektfelder einführen.

Exkurs: Entwurfsmuster „Adapter“



Ziel (Target) Das Ziel definiert die Schnittstelle, die der Klient nutzen kann.

Klient (Client) Der Klient nutzt Dienste über inkompatible Schnittstellen und greift dabei auf adaptierte Schnittstellen zurück.

Dienst (Adaptee) Der Dienst bietet wiederzuverwendende Dienstleistungen mit fest definierter Schnittstelle an.

Adapter Der Adapter adaptiert die Schnittstelle des Dienstes auf die Schnittstelle zum Klienten.

```java
/**
 * GoF: Adaptee
 */
public class Dienst {

  public void agiereSpeziell() {
      System.out.println("Agiere speziell!");
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Dienst.java`

```java
public interface Ziel {
  public void agiere();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Ziel.java`

```java
public class Adapter extends Dienst implements Ziel {

  @Override
  public void agiere() {
      System.out.print("agiere: ");
      agiereSpeziell();
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Adapter.java`

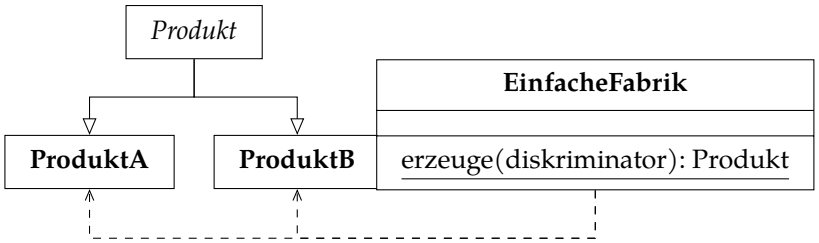
```java
public class Klient {

  public static void main(String[] args) {
    new Adapter().agiere();
    // agiere: Agiere speziell!
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Klient.java`

(c) Verwenden Sie das Entwurfsmuster **Simple Factory**, um die Erzeugung von Angestellten, Kunden, sowie Adapter-Instanzen aus Aufgabe (b) zu vereinheitlichen. Die entsprechende Erzeugungs-Methode soll neben einem Typ-Diskriminator (z. B. einem Aufzählungstypen oder mehreren booleschen Werten) alle Parameter übergeben bekommen, die für den Konstruktor irgendeiner Implementierungsklasse des Interface `Person` notwendig sind.

Hinweis: Um eine Adapter-Instanz zu erzeugen, müssen Sie möglicherweise zwei Konstruktoren aufrufen.

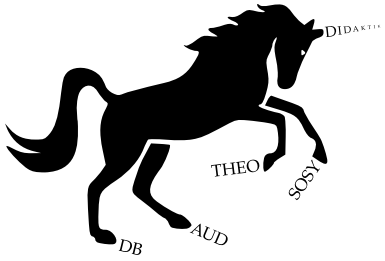
Exkurs: Entwurfsmuster „Einfache Fabrik“



EinfacheFabrik Eine Klasse mit einer Erzeugunsmethode, die über eine größere Bedingung verschiedene Objekt instanziert.

Produkt Eine abstrakte Klasse, die von den konkreten Produkten geerbt wird.

KonkretesProdukt Ein konkretes Produkt, das von der einfachen Fabrik erzeugt wird.



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.Der TEX-Quelltext dieses Dokuments kann unter folgender URL aufgerufen werden: `https://github.com/bschlangaul-sammlung/examens-aufgaben/blob/main/Staatsexamen/66116/2015/03/Thema-1/Teilaufgabe-2/Aufgabe-3.tex`