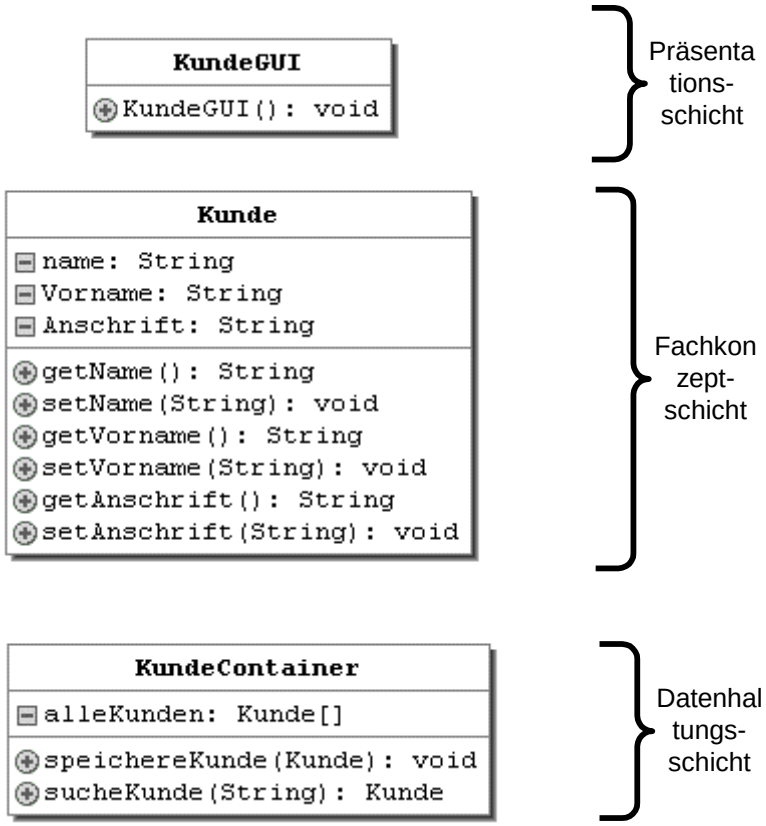
1 Strukturierung

Ein grundlegendes Prinzip beim Entwurf von Software-Systemen besteht heute in der klaren Trennung zwischen *Benutzungsoberfläche*, *Fachkonzept* und *Datenhaltung*. Diese Art der Strukturierung bezeichnet man als 3-Schicht-Architektur.

Die Programmfunktionalität wird in entsprechenden Klassen programmiert, die dann der jeweiligen Schicht zugeordnet werden. Es handelt sich dabei um eine virtuelle Zuordnung, die im Programmcode nicht ersichtlich ist.



Vorteile liegen in der besseren Änderbarkeit, Wartbarkeit und Übersichtlichkeit, außerdem werden die Abhängigkeiten innerhalb des Systems verringert.

1.1 Präsentationsschicht

Die Präsentationsschicht stellt i.d.R. die Verbindung zur Benutzeroberfläche her. Sie wird auch als *Presentation-Layer*, *GUI-Schicht*, *Client Tier* oder *Front-End* bezeichnet.

Die Klassen der Präsentationsschicht sollen immer mit dem Suffix GUI versehen werden ("GUI-Klassen"). Das Präfix wird gebildet aus dem Namen der Fachkonzeptklasse, also z.B. KundeGUI, wenn die Fachkonzeptklasse Kunde heißt.

Aufgaben der GUI-Klasse:

- Aufbau der Benutzungsoberfläche
- Eingaben der Benutzer lesen
- Ergebnisse ausgeben
- Referenzen auf die erzeugten Objekte merken, damit später Datenänderungen vorgenommen werden können.

Anmerkung:
Üblich ist die Vorgehensweise, dass auch von der GUI-Klasse ein Objekt erzeugt wird. Dies geschieht in der Methode `main`.
Daraus folgt, dass es von der GUI-Klasse einen Konstruktor geben muss. In diesem werden dann alle Anweisungen geschrieben, die eigentlich in der Methode `main` stehen würden.
Beispiel:

```
public class KundeGUI
{
  private Kunde einKunde;

  public KundeGUI()
  {
    einKunde = new Kunde();
    einKunde.setName("Maier");
  ...
  }

  public static void main(String [] arg)
  {
    KundeGUI eineGUI = new KundeGUI();
  }
}
```

Somit kann main dann auch in eine eigene Start-Klasseausgelagert werden, sodass KundeGUI eine reine GUI-Klasse wird.

```
public class KundeGUI
{
  private Kunde einKunde;

  public KundeGUI()
  {
     einKunde = new Kunde();
     einKunde.setName("Maier");
  ...
  }
}

public class StartKunde
{
  public static void main(String [] arg)
  {
    KundeGUI eineGUI = new KundeGUI();

}
```

1.2 Fachkonzeptschicht

Eine Fachkonzeptschicht realisiert die im Pflichtenheft festgelegten fachlichen Anforderungen, also z.B. Kunde, Lieferant. Sie enthält die eigentliche Anwendungslogik. Daher wird die Fachkonzeptschicht auch als *Anwendungslogikschicht, Application-Layer, Application-Server Tier, Businessschicht, Middle Tier oder Enterprise Tier* bezeichnet.

Es wird jedoch noch mindestens eine weitere Klasse i.d.R. eine GUI-Klasse benötigt, von der aus Botschaften verschickt werden, um

- Objekte von der Fachkonzeptklasse zu erzeugen und
- die erzeugten Objekte zu manipulieren

1.3 Datenhaltungsschicht

In Datenhaltungsklassen werden die erzeugten Objekte einer Fachkonzeptklasse verwaltet. Sie werden auch als *Container-Klassen, Data-Server Tier* oder *Back End* bezeichnet.
Die Namensgebung erhält den Suffix „*Container*“, also z.B. KundenContainer. In diesen Klassen befinden sich die Methoden, um Objekte persistent zu speichern.
Die GUI-Klasse enthält eine Referenz auf die Containerklasse, damit die Instanzmethoden von dort verwendet werden können.
So bleibt die Art der Datenhaltung (DB, File, Vector, ...) vor der Anwendung verborgen.

2 Kommunikation

Der Aufruf der einzelnen Schichten läuft immer von oben nach unten ab. Klassen aus den "oberen" Schichten dürfen nur Klassen aus den "unteren" Schichten aufrufen, nicht umgekehrt! Die Fachkonzeptklassen dürfen nicht direkt mit dem GUI-System kommunizieren, sprich Methoden aus GUI-Klassen aufrufen.

Ebenso dürfen Methoden aus Containerklassen keine Methoden aus GUI-, oder Fachkonzeptklassen aufrufen.