

## 6 Observer-Muster

In der Vorlesung wurde eine Wetterstation als Beispiel für das Observer Muster gezeigt. Analog dazu soll ein System zur Verwaltung von Zeitungen implementiert werden. Dabei können sich verschiedene Abonnenten bei einem Zeitungsverlag registrieren, um die Zeitung zu erhalten. Erstellen Sie zunächst eine Klasse **Zeitung** mit den folgenden Attributen:

```
private Date datum;  
private String titel;
```

Implementieren Sie die benötigten Getter- und Setter-Methoden, sowie einen Konstruktor.

Schreiben Sie eine Schnittstelle **IAbonnant**, die eine Methode `erhalteZeitung` deklariert:

```
public interface IAbonnant {  
    public void erhalteZeitung(Zeitung zeitung);  
}
```

Schreiben Sie eine Klasse **Abonnant**. Die Klasse soll die Schnittstelle **IAbonnant** implementieren. Außerdem soll ein Abonnant durch einen Namen (vom Typ String) definiert sein. Implementieren Sie die benötigten Getter- und Setter-Methoden und einen Konstruktor. Die Methode `erhalteZeitung` ist exemplarisch und soll lediglich einen Text wie folgt ausgeben:

```
System.out.println(this.name + " hat die Zeitung " +  
    ↳ zeitung.getTitel() + " am " + zeitung.getDatum() + " erhalten.");
```

Schreiben Sie eine Klasse **Zeitungsverlag**. Die Klasse muss Methoden zum Registrieren, Entfernen und Benachrichtigen von Abonnenten anbieten. Die Abonnenten werden in eine `ArrayList<IAbonnent>` `abonnenten` gespeichert:

```
private ArrayList<IAbonnent> abonntenen = new ArrayList<IAbonnent>();

public void registriereBeobachter(IAbonnent abonnent) {
    if (!this.abonntenen.contains(abonnent))
        abonntenen.add(abonnent);
}

public void entferneBeobachter(IAbonnent abonnent) {
    abonntenen.remove(abonnent);
}

public void benachrichtigeBeobachter(Zeitung zeitung) {
    for (IAbonnent abonnent : this.abonntenen) {
        abonnent.erhalteZeitung(zeitung);
    }
}
```

Fügen Sie eine Methode `verteileZeitung` in `Zeitungsverlag` hinzu:

```
public void verteileZeitung(String titel) {
    Zeitung zeitung = new Zeitung(new Date(), titel);
    benachrichtigeBeobachter(zeitung);
}
```

Schreiben Sie eine Testklasse, um die Funktionalität Ihres Programms zu überprüfen. Erstellen Sie einen Zeitungsverlag und mehrere Abonnenten. Rufen Sie die Methode `verteileZeitung` von dem Verlag auf.

```
Zeitungsverlag elsevier = new Zeitungsverlag();
IAbonnent infBI = new Abonnent("Lehrstuhl fuer Informatik im
↳ Bauwesen");
IAbonnent max = new Abonnent("Max Mustermann");
elsevier.registriereBeobachter(infBI);
elsevier.registriereBeobachter(max);
elsevier.verteileZeitung("Java Design Patterns");
```

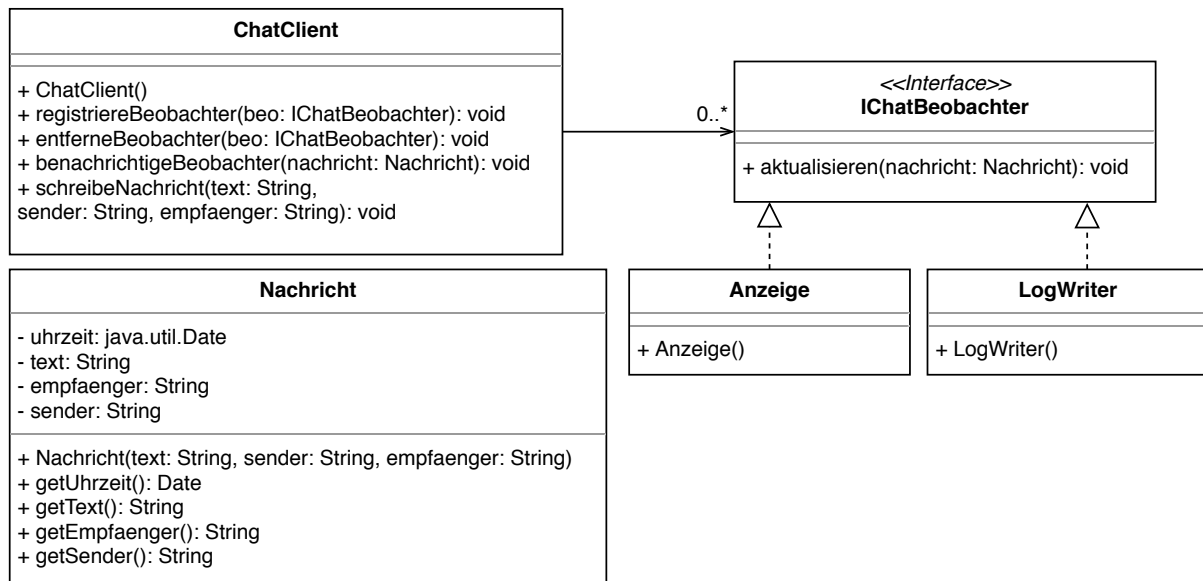


Abbildung 1: UML-Klassendiagramm des Chat-Clients

## Aufgaben

1. Zeichnen Sie ein Klassendiagramm für die Klassen **Zeitung**, **Abonnent**, **IAbonnent** und **Zeitungsverlag**. Orientieren Sie sich dabei an dem Beispiel aus der Vorlesung.
2. Es soll ein Chat-Client auf Basis des Observer-Musters implementiert werden. Orientieren Sie sich dabei an dem UML-Klassendiagramm in Abb. 1.
3. Lassen Sie Nachrichten, die von Objekten der Klasse **Anzeige** empfangen werden, auf der Console mittels `System.out.println()` ausgeben. Der eigentlichen Nachricht soll dabei das aktuelle Datum mit Uhrzeit und der Name des Senders und Empfängers vorangestellt sein.
4. Objekte der Klasse **LogWriter** nutzen einen **FileWriter**, um erhaltene Nachrichten auf der Festplatte abzuspeichern. Das Format soll dabei auch das aktuelle Datum und die Uhrzeit, sowie den Namen des Senders enthalten. Erzeugen Sie pro Empfänger eine Datei, in der alle Nachrichten des Benutzers gespeichert werden.
5. Implementieren Sie eine Testklasse **ChatTest**, in der zunächst ein **ChatClient** erzeugt wird und anschließend zwei Beobachter vom Typ **Anzeige** und **LogWriter**. Versenden Sie anschließend einige Nachrichten, um die Funktionalität zu testen.



**Tipp**

Damit die erzeugten .txt Dateien in Eclipse sichtbar werden, klicken Sie auf den Projektnamen im Project Explorer und drücken Sie die Taste **F5** für „Aktualisieren“