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 **`IAbonnent`**, die eine Methode `erhalteZeitung` deklariert:

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

Schreiben Sie eine Klasse **`Abonnent`**. Die Klasse soll die Schnittstelle **`IAbonnent`** implementieren. Außerdem soll ein Abonnent 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> abonnenten = new ArrayList<IAbonnent>();

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

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

public void benachrichtigeBeobachter(Zeitung zeitung) {
    for (IAbonnent abonnent : this.abonnenten) {
        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");
```

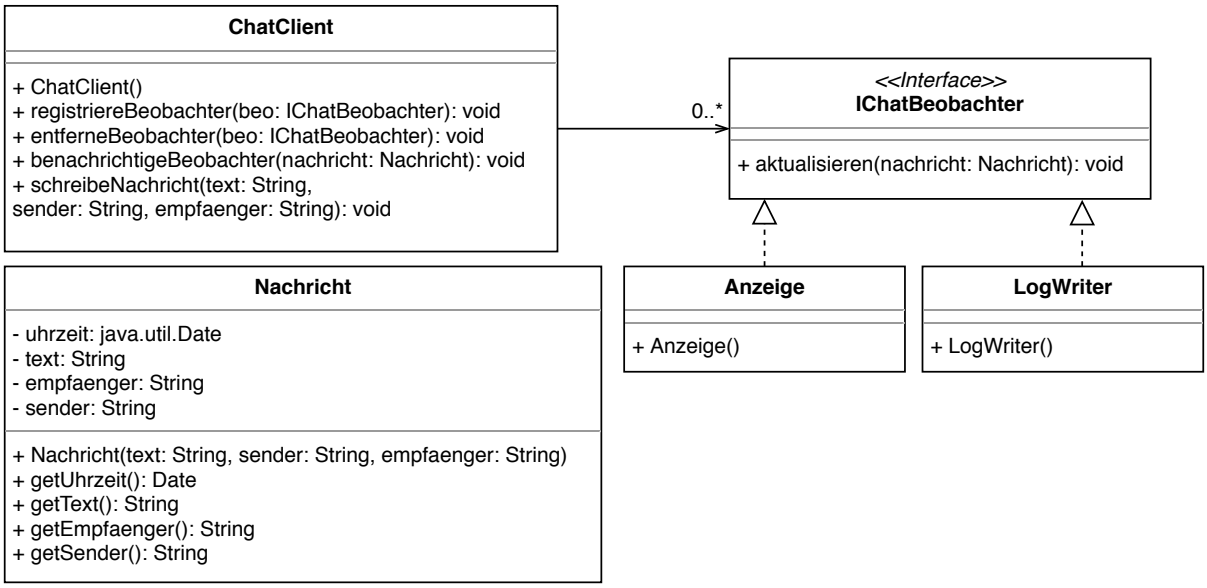
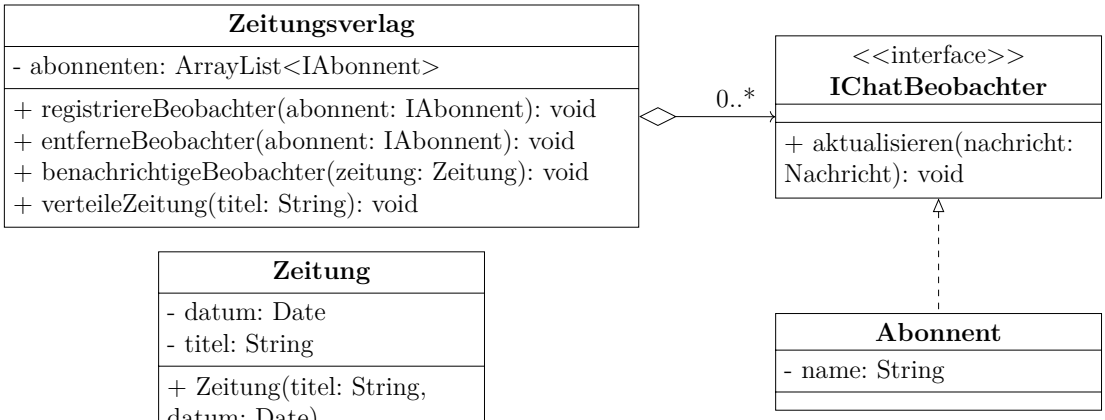


Abbildung 1: UML-Klassendiagramm des Chat-Clients

Aufgaben

1. Zeichnen Sie ein Klassendiagramm für die Klassen **`Zeitung`**, **`Abonnent`**, **`IAbonnent`** und **`ZeitungsVerlag`**. Orientieren Sie sich dabei an dem Beispiel aus der Vorlesung.

Lösung:



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“

Lösung:

Nachricht.java

```
package chatclient;

import java.util.Date;

public class Nachricht {

    private Date uhrzeit;
    private String text;
    private String empfaenger;
    private String sender;

    public Nachricht(String text, String sender, String empfaenger) {
        uhrzeit = new Date();  // Jetzige Uhrzeit
        this.text = text;
        this.sender = sender;
        this.empfaenger = empfaenger;
    }

    public Date getUhrzeit() {
        return uhrzeit;
    }
    public String getText() {
        return text;
    }
    public String getEmpfaenger() {
        return empfaenger;
    }
    public String getSender() {
```

```
        return sender;
    }
}
```

ChatClient.java

```
package chatclient;

import java.util.ArrayList;

public class ChatClient {
    ArrayList<IChatBeobachter> beobachter = new
↪   ArrayList<IChatBeobachter>();

    public void registriereBeobachter(IChatBeobachter beo) {
        beobachter.add(beo);
    }
    public void entferneBeobachter(IChatBeobachter beo) {
        beobachter.remove(beo);
    }
    public void benachrichtigeBeobachter(Nachricht nachricht) {
        for(IChatBeobachter b: beobachter)
            b.aktualisieren(nachricht);
    }
    public void schreibeNachricht(String text, String sender, String
↪   empfaenger) {
        benachrichtigeBeobachter(new Nachricht(text, sender,
↪   empfaenger));
    }
}
```

IChatBeobachter.java

```
package chatclient;

public interface IChatBeobachter {

    public void aktualisieren(Nachricht nachricht);
}
```

Anzeige.java

```
package chatclient;

public class Anzeige implements IChatBeobachter {
```

```
    @Override
    public void aktualisieren(Nachricht nachricht) {
        System.out.println("[" + nachricht.getUhrzeit() + "] Sender: "
↪   + nachricht.getSender() + ", Empfänger: " +
↪   nachricht.getEmpfaenger() + " - " + nachricht.getText());
    }

}
```

LogWriter.java

```
package chatclient;

import java.io.FileWriter;
import java.io.IOException;

public class LogWriter implements IChatBeobachter {

    @Override
    public void aktualisieren(Nachricht nachricht) {
        String dateiname = nachricht.getEmpfaenger() + ".txt";
        try {
            FileWriter fw = new FileWriter(dateiname, true);
            fw.append("[" + nachricht.getUhrzeit() + "] " +
↪   nachricht.getSender() + ": " + nachricht.getText() + "\n");
            fw.close();
        } catch (IOException e) {
            System.out.println("Datei " + dateiname + " konnte nicht
↪   geöffnet werden");
            e.printStackTrace();
        }
    }

}
```

ChatTest.java

```
package chatclient;

public class ChatTest {
    public static void main(String[] args) {
        ChatClient chat = new ChatClient();
        chat.registriereBeobachter(new LogWriter());
        chat.registriereBeobachter(new Anzeige());

        chat.schreibeNachricht("Hallo", "Alice", "Bob");
```

```
        chat.schreibeNachricht("Nein", "Bob", "Alice");
        chat.schreibeNachricht(":(", "Alice", "Bob");
    }
}
```