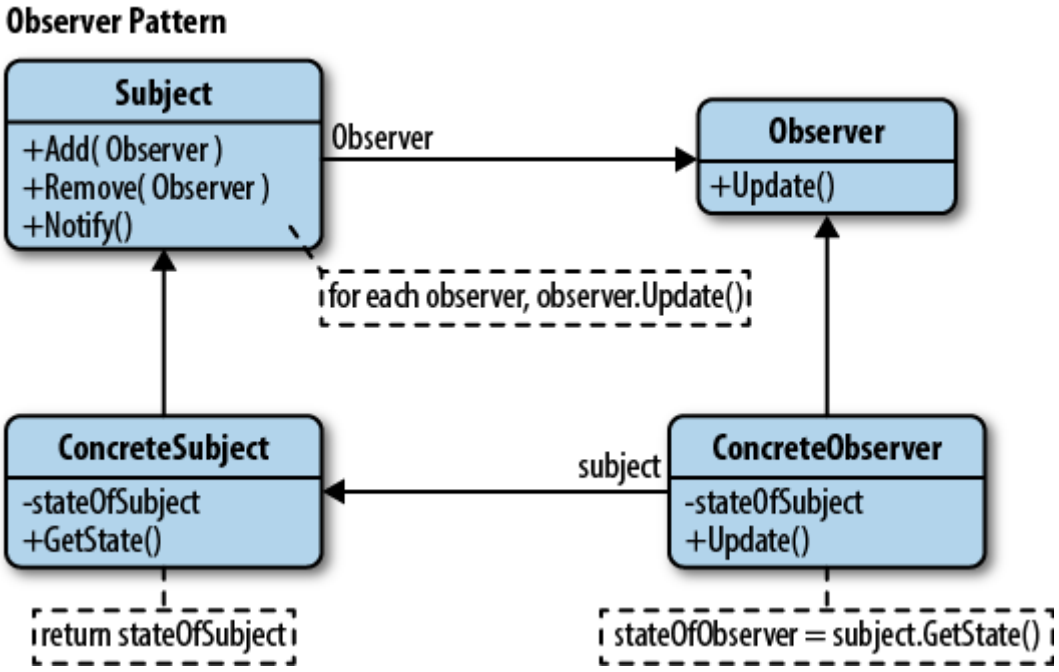
# Seminar 6




1. Erstellen Sie eine einfache Wetterstationsanwendung, bei der Wetterdaten von mehreren Anzeigen beobachtet werden. Erstellen Sie drei Anzeigen-Klassen: CurrentConditionsDisplay, StatisticsDisplay und ForecastDisplay. Diese Anzeigen sollten Änderungen in den Wetterdaten beobachten.

Erstellen Sie eine WeatherData Klasse, die als das Subjekt dient. Diese sollte Methoden zum Registrieren, Entfernen und Benachrichtigen von Observers sowie Methoden zum Festlegen der Wetterdaten (z. B. Temperatur, Luftfeuchtigkeit, Luftdruck) haben.

Wenn sich die Wetterdaten ändern, benachrichtigen Sie alle registrierten Displays, damit sie sich mit den neuesten Daten aktualisieren.

1. Definieren Sie eine Schnittstelle, Observer, mit einer update Methode, die Wetterdaten als Parameter akzeptiert.
2. Erstellen Sie die Display Schnittstelle mit einer display Methode.
3. Implementieren Sie die Klassen CurrentConditionsDisplay, StatisticsDisplay und ForecastDisplay. Jede dieser Klassen sollte die Schnittstellen Observer und Display implementieren.
4. Implementieren Sie die Klasse WeatherData, die eine Liste von Observer und die Wetterdaten (Temperatur, Luftfeuchtigkeit, Luftdruck) enthält. Sie sollte Methoden zum Registrieren, Entfernen und Benachrichtigen von Beobachtern haben.
5. In der Klasse WeatherData sollten Sie, wenn sich die Wetterdaten ändern (z. B. Temperatur, Luftfeuchtigkeit oder Luftdruck), die Methode notifyObservers aufrufen, um alle registrierten Observers über die Aktualisierungen zu informieren.

6. In der update Methode der Observers sollten Sie die Wetterdaten aktualisieren und anzeigen, wenn Sie benachrichtigt werden.
7. Erstellen Sie eine Main-Klasse, um das Observer-Pattern zu testen. In der Main-Klasse können Sie Instanzen der Wetterdaten, Anzeigen erstellen, die Displays als Beobachter der Wetterdaten registrieren und die Wetterdaten ändern, um die Aktualisierungen in den Observer Klassen zu beobachten.

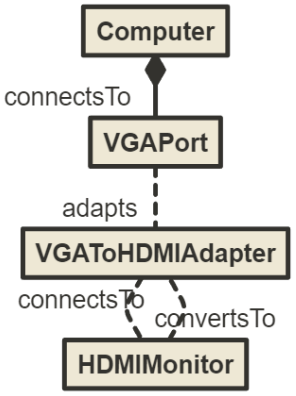
```
public class Main {
    public static void main(String[] args) {
        WeatherData weatherData = new WeatherData();

        CurrentConditionsDisplay currentConditionsDisplay = new
CurrentConditionsDisplay(weatherData);
        StatisticsDisplay statisticsDisplay = new StatisticsDisplay(weatherData);
        ForecastDisplay forecastDisplay = new ForecastDisplay(weatherData);

        // Simulate changes in weather data
        weatherData.setMeasurements(75, 60, 30.4f);
        weatherData.setMeasurements(80, 65, 29.2f);
        weatherData.setMeasurements(72, 55, 29.5f);
    }
}
```

---

2 Simulieren Sie die Verwendung eines VGA-zu-HDMI-Adapters, um einem Computer mit einem VGA-Port die Verbindung mit einem HDMI-Monitor zu ermöglichen und so das Adaptermuster zu veranschaulichen.




1. Erstellen Sie zwei Klassen: Computer und HDMIMonitor. Die Klasse Computer stellt einen Computer mit einem VGA-Port dar, und die Klasse HDMIMonitor stellt einen HDMI-Monitor dar.
2. Implementieren Sie eine Schnittstelle VGAPort mit einer Methode connectVGA(). Die VGAPort-Schnittstelle soll die VGA-Verbindung am Computer repräsentieren.
3. Erstellen Sie eine Klasse namens VGAToHDMIAdapter, die die VGAPort-Schnittstelle implementiert. Diese Adapterklasse sollte den VGA-Port an eine HDMI-Verbindung anpassen.
4. Implementieren Sie eine Methode connectHDMI() in der Klasse HDMIMonitor, um die Verbindung zu einem HDMI-Monitor zu simulieren.
5. Erstellen Sie eine User-Klasse, die einen Computer mit einem HDMI-Monitor verbindet. Die User-Klasse sollte die VGAPort-Schnittstelle verwenden, um den Computer über den Adapter mit dem HDMI-Monitor zu verbinden.