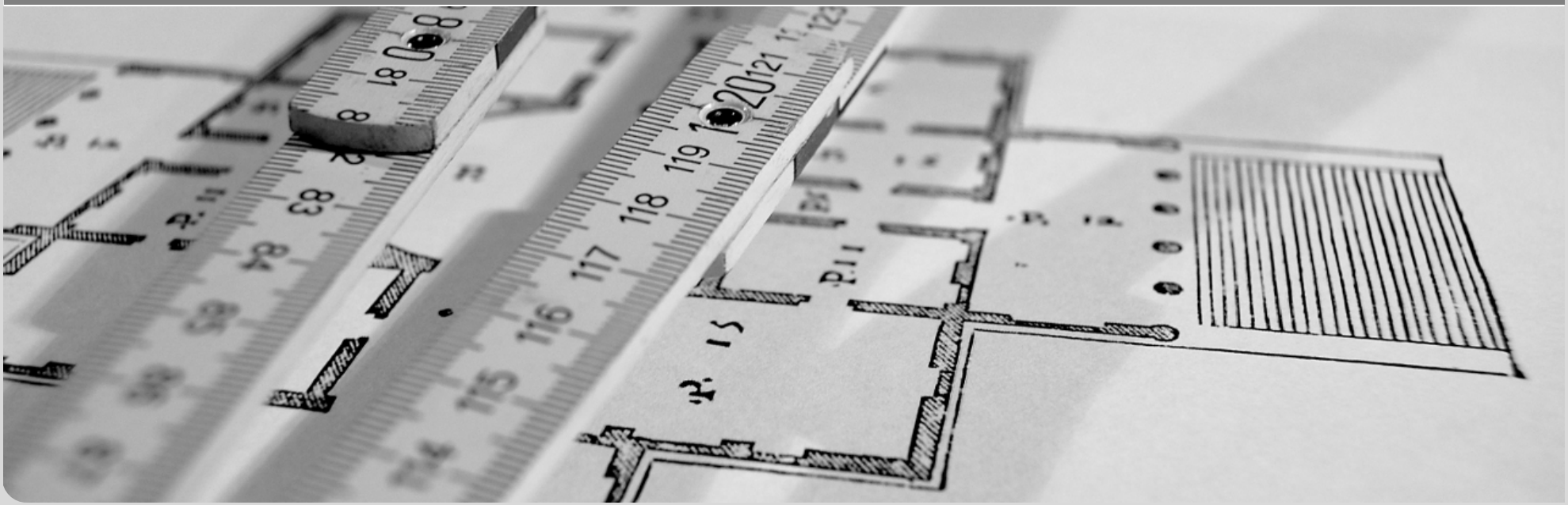
Karlsruher Institut für Technologie

Software Design and Quality

SWT
3. Tutorium

Tino Fuhrmann | 2. Juni 2015



Gliederung

3. Übungsblatt

- Prüfung Scheinkriterium an richtiger Stelle
- git commits nicht vergessen
- Menu korrekt einfügen
- Historie im Automaten mit modellieren und nicht einfach weglassen
- STAT ist in OMN/PARS bereits kodiert
- Korrektes UML verwenden und genau nach Vorlesung übersetzen

Schichtenarchitektur

Beschreibung?

Schichtenarchitektur

Beschreibung?
Vorteile und Nachteile?

Fassade

Nutzen?

Wichtig: MVC

- Modell: Verantwortlich für Haltung der Anwendungsdaten
- View: Verantwortlich für Darstellung der Daten
- Steuerung: Verantwortlich für Benutzerinteraktion/Updatet View

Fördert Austauschbarkeit der einzelnen Komponenten.

Fließband

- Abarbeitung von Daten in unterschiedlichen Stufen
- Vorteil: einfache Parallelisierbarkeit

Framework

- bietet nahezu vollständiges Programm
- einfache Erweiterbarkeit durch geplante Lücken, die durch Software gefüllt werden können.

Einführung

Wieso werden Entwurfsmuster verwendet?

Einführung

Wieso werden Entwurfsmuster verwendet?

- Verbessern Kommunikation im Team
- Muster erfassen wesentliche Konzepte und bringen sie in verständliche Form
- Muster dokumentieren und fördern Stand der Kunst
  - Helfen weniger erfahrenen Entwicklern
  - Vermeiden Neuerfindung des Rades
- Muster können Code-Qualität und Struktur verbessern

Aufgabe

- Jeder bekommt eine Karte mit einem Entwurfsmuster
- Aufgabe: Kurz überlegen und dann in Kleingruppen mit Beispiel vorstellen.
- Einteilung: gleich.

Aufgabe

- Jeder bekommt eine Karte mit einem Entwurfsmuster
- Aufgabe: Kurz überlegen und dann in Kleingruppen mit Beispiel vorstellen.
- Einteilung: gleich.
- Gruppeneinteilung: Durchzählen bis x.

Feedback

Wie fandet ihr diese Aufgabenstellung?
Beibehalten für die nächsten Entwurfsmuster?
Eher nur Klausuraufgaben dazu?

1. Aufgabe

In den Java-Bibliotheken lassen sich einige Entwurfsmuster finden. Identifizieren Sie anhand der Zitate aus der Dokumentation, welche das sind. Begründen Sie Ihre Aussage gut, sonst bekommen Sie keine Punkte. Geben Sie insbesondere an, welche Klasse, Methode, Instanz, usw. welche Rolle in dem von Ihnen angegebenen Muster einnehmen soll!

Klasse java.util.Arrays (JDK Version 1.3). (3P)

`public static java.util.List asList(Object[] a)`

Returns a fixed-size list backed by the specified array. (Changes to the returned list "write through" to the array.) This method acts as bridge between array-based and collection-based APIs, in combination with Collection.toArray. The returned list is serializable.

Parameters: a - the array by which the list will be backed.

Returns: a list view of the specified array.

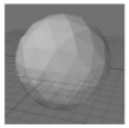
See Also: Collection.toArray()

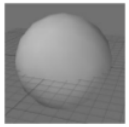
Lösung

Muster: Adapter
a ist das adaptierte Projekt
Rückgabewert ist Adapter.

2. Aufgabe: Brücke

Aufgabe 2: Entwurfsmuster (Brücke und Beobachter) (12P)

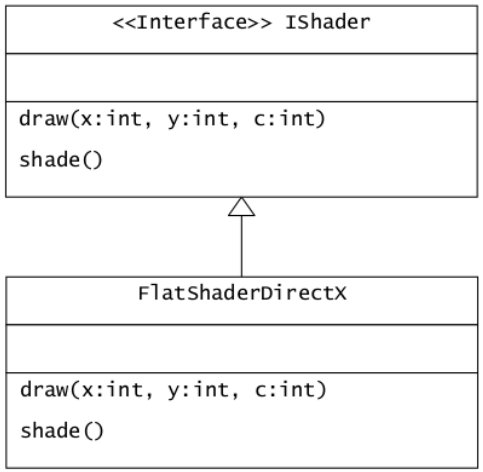




Der Begriff **Shading** bezeichnet in der 3D-Computergrafik im allgemeinen Sinne die Simulation der Oberfläche eines Objekts. Für die Berechnung der Oberfläche gibt es viele verschiedene Verfahren – unter anderem das **Flat Shading** (Abb. links oben) und das **Gouraud Shading** (Abb. links unten).

Sie haben einen Flat Shader für die DirectX-Programmierschnittstelle entsprechend dem folgenden UML-Klassendiagramm entworfen.

Die Methode **`shade()`** in der Klasse **`FlatShaderDirectX`** implementiert den Flat Shading-Algorithmus und verwendet die Methode **`draw(x:int, y:int, c:int)`**, um einen Punkt unter Verwendung der DirectX-Programmierschnittstelle zu zeichnen.

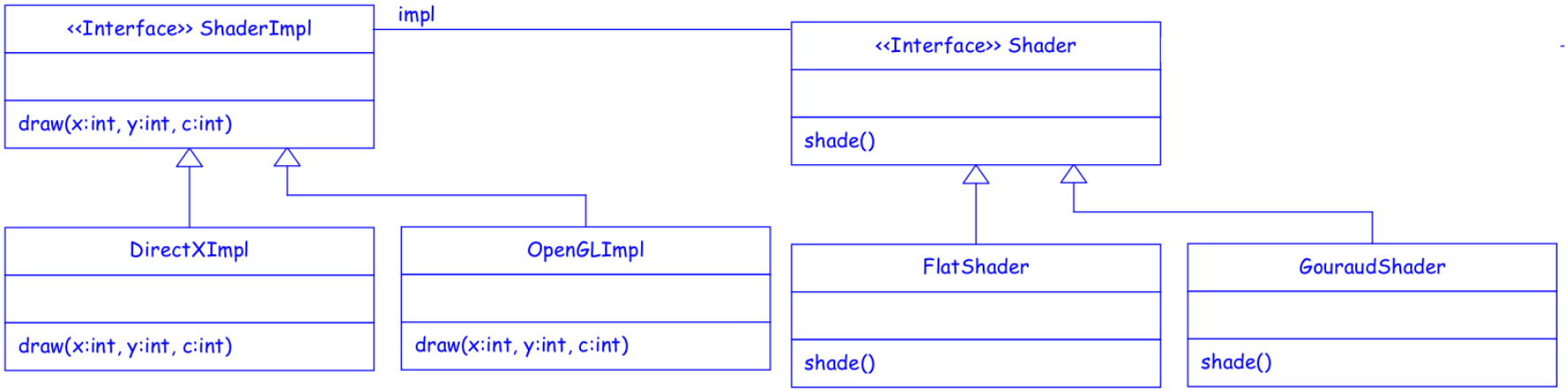


Neben Flat Shading möchten Sie zusätzlich noch das Gouraud Shading-Verfahren anbieten. Außerdem möchten Sie beide Shading-Verfahren sowohl für die DirectX-API als auch für die OpenGL-Programmierschnittstelle implementieren.

a.) Verwenden Sie das Entwurfsmuster **Brücke** um die Abstraktion (Methode **`shade`**) von der Implementierung (Methode **`draw`**) zu trennen.

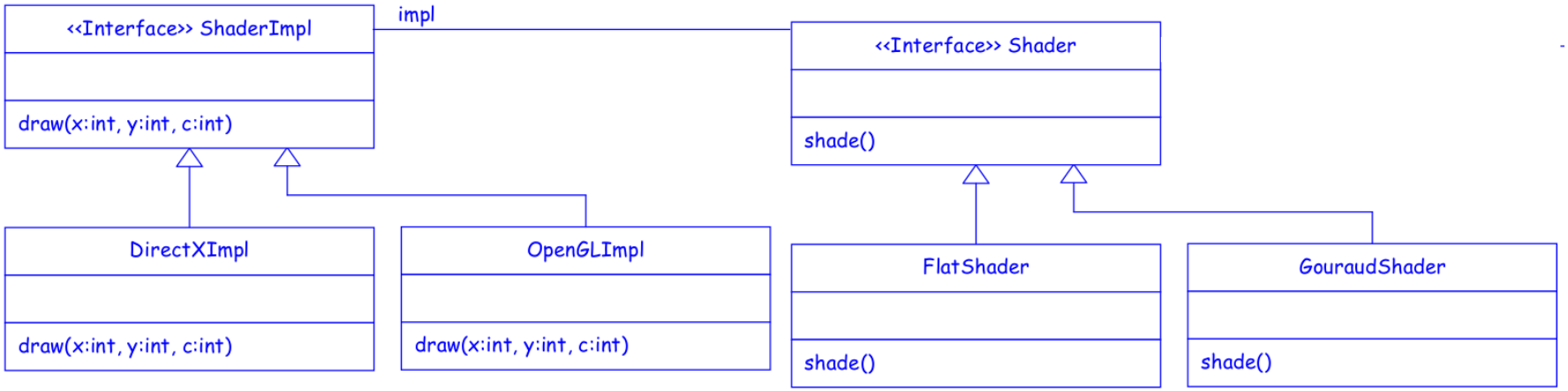
Hinweis: Trennen Sie die obige Schnittstelle **`IShader`** geeignet auf und erweitern Sie Ihr UML-Klassendiagramm um die konkreten Klassen **`DirectXImpl`** und **`OpenGLImpl`**, **`FlatShader`** und **`GouraudShader`**. Tragen Sie Vererbungsbeziehungen und Assoziationen entsprechend dem Entwurfsmuster Brücke ein. (7P)

2. Aufgabe: Lösung



Geben Sie nachfolgend in Java-Notation an wie – entsprechend dem geforderten Ent- wurf mit dem Entwurfsmuster Brücke – die Methode shade im FlatShader einen schwarzen Punkt (c = 0) mit den Koordinaten x=23 und y=42 zeichnen kann.

2. Aufgabe: Lösung



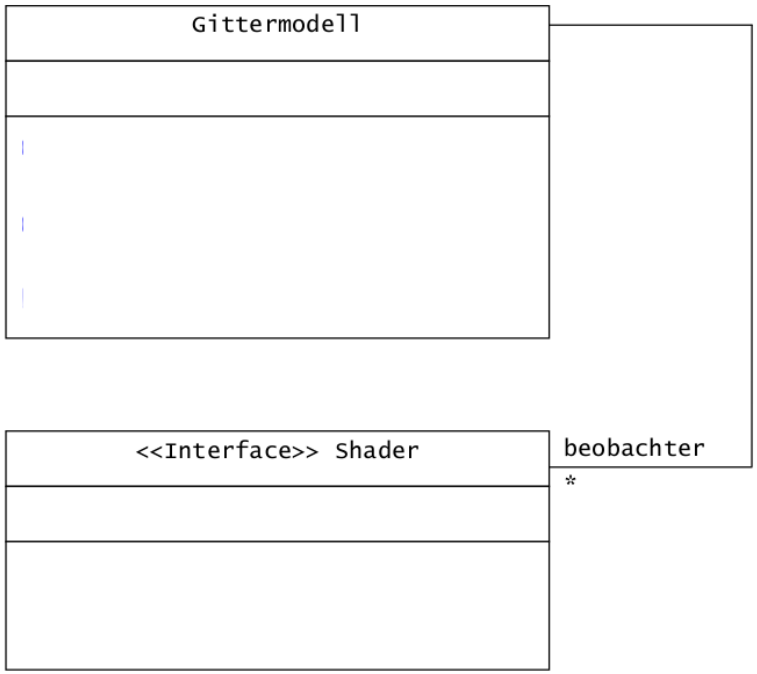
Geben Sie nachfolgend in Java-Notation an wie – entsprechend dem geforderten Ent- wurf mit dem Entwurfsmuster Brücke – die Methode shade im FlatShader einen schwarzen Punkt (c = 0) mit den Koordinaten x=23 und y=42 zeichnen kann.

impl.draw(23, 42, 0);

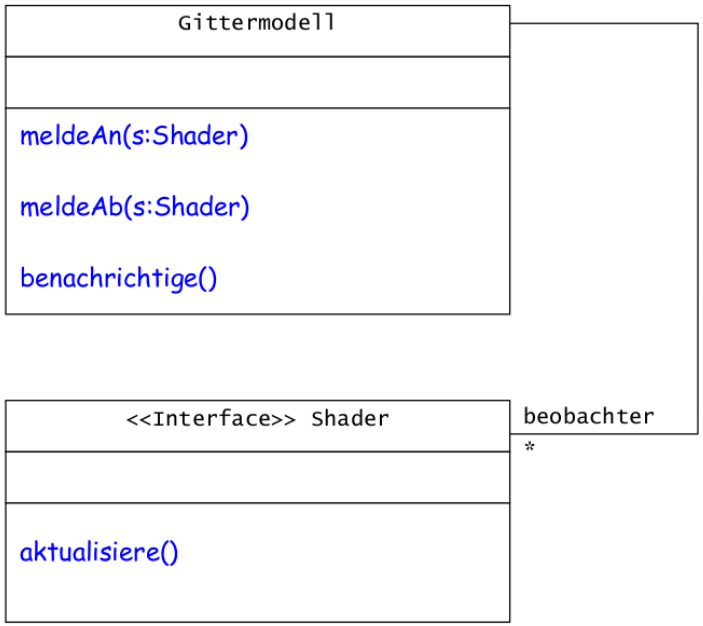
2. Aufgabe: Teil 2

c.) Das Gittermodell, für das Ihre Shader-Klassen die Darstellungen der Oberflächen berechnen, wird von der Klasse **`Gittermodell`** verwaltet. Bei jeder Änderung des Gittermodells (Subjekt) soll der verwendete **`Shader`** (Beobachter) benachrichtigt werden. Verwenden Sie das Entwurfsmuster **Beobachter** und erweitern Sie das folgende UML-Klassendiagramm um Methoden-Signaturen für das An- und Abmelden, Benachrichtigen und Aktualisieren des Beobachters. (3P)

Hinweis: Definieren Sie keine zusätzlichen abstrakten Subjekt- und Beobachterklassen sondern verwenden und erweitern Sie lediglich die gegebene Schnittstelle **`Shader`** und die Klasse **`Gittermodell`**.



2. Aufgabe: Teil 2 Lösung



Geben Sie eine Java-Datenstruktur für die beobachter-Assoziation in obigem UML- Klassendiagramm an. Geben Sie für diese Datenstruktur in Java-Notation die Implementierung für das Benachrichtigen angemeldeter Shader an.

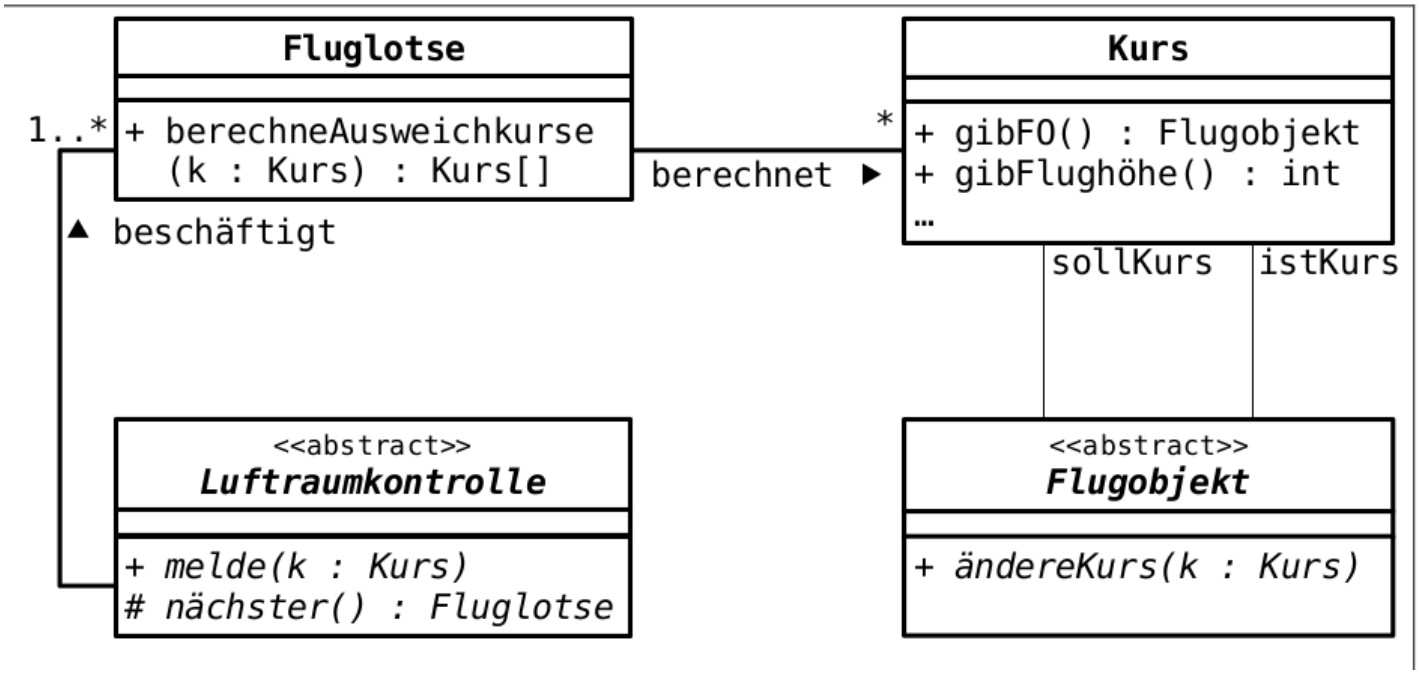
2. Aufgabe: Implementierung

Geben Sie eine Java-Datenstruktur für die beobachter-Assoziation in obigem UML- Klassendiagramm an. Geben Sie für diese Datenstruktur in Java-Notation die Implementierung für das Benachrichtigen angemeldeter Shader an.

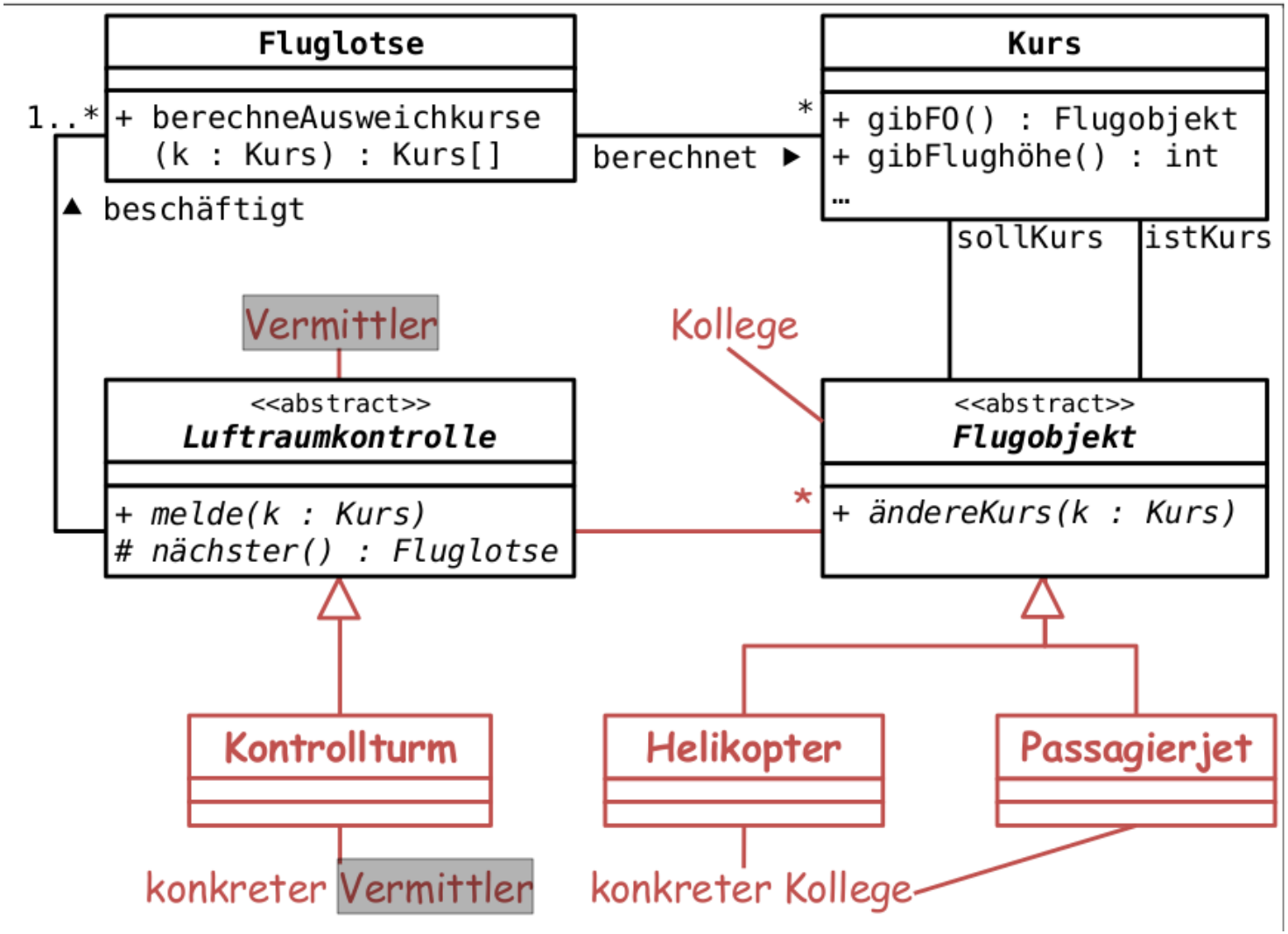
```
Vector beobachter;
for (Shader s : beobachter) {
s.aktualisiere();
}
```

Aufgabe 3: Vermittler

Orientieren Sie sich am Vermittler-Entwurfsmuster, um das Klassendiagramm um die Klas- sen Kontrollturm (eines Flughafens) sowie Helikopter und Passagierjet zu ergänzen. Ge- ben Sie Multiplizitäten an, sofern diese ungleich 1 sind. Geben Sie für die Klassen Luf t-raumkontrolle , Flugobjekt , Kontrollturm , Helikopter und Passagierjet die jeweilige Rolle im Vermittler-Entwurfsmuster an.



Aufgabe 3: Lösung



Feedback

Wie fandet ihr die Tutorien bis jetzt?
Änderungswünsche?