



http://ovm-kassel.de Information	
Information IT-AE-UML-INFO-3.3 Objektorientierte Modellierung (OOM)	
Code	IT-AE-UML-INFO-3.3
Autor	André Bauer <a(dot)bauer(at)ovm-kassel(dot)de>
Datum	25. Februar 2018
Links	Wikipedia: Objektorientierte Analyse und Design
Verwandte Literatur	IT-AE-UML-INFO-3.1
Lernjobs	IT-AE-UML-LJ-3.1 bis 3.4
Lizenz	 Dieses Werk ist lizenziert unter einer Creative Commons Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz .

Objektorientierte Modellierung (OOM)

In der objektorientierten Modellierung werden u. a. die bereits vorgenommenen Modellierungen in den Klassendiagrammen verfeinert und z. B. Datentypen, initiale Werte und weitere Methoden eingefügt. Gegebenenfalls werden die Diagramme um weitere Klassen ergänzt, die sich nicht unmittelbar aus der Analyse ergeben haben, aber für die Implementierung in der gewählten Programmiersprache benötigt werden, wie die Definition einer Aufzählung (englisch Enumeration).

1. Enumerations

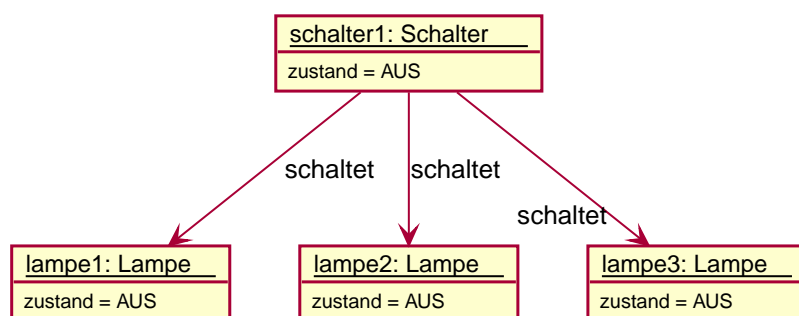


Abbildung 1. Objektdiagramm für einen Aufbau mit einem Schalter und drei Lampen zu Beginn der User-Story-1

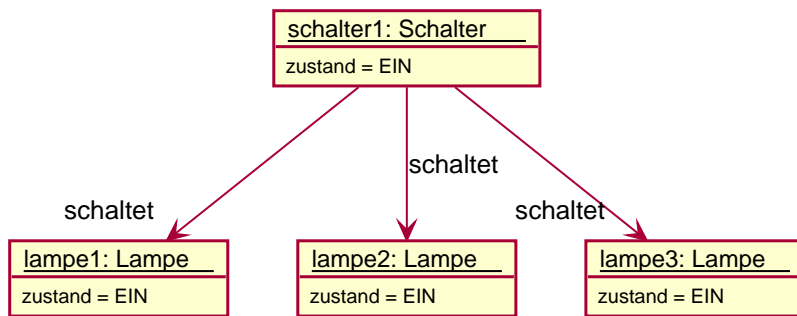


Abbildung 2. Objektdiagramm für einen Aufbau mit einem Schalter und drei Lampen zum Ende der User-Story-1

In den Objektdiagrammen in [Abbildung 1](#) und [Abbildung 2](#) haben die Objekte das Attribut **zustand** mit den möglichen Werten **ON** und **OFF**. Anstatt eines vorhandenen Java-Datentyps wie **int**, **boolean** oder **string** erstellen wir mithilfe einer Aufzählung (engl. enumeration, in Java abgekürzt zu **enum**) einen passenden neuen und aussagekräftigen Datentyp. Eine Enumeration ermöglicht es, einen Datentyp durch die Aufzählung der möglichen Werte zu definieren.

Quellcode 1. Enumertation State in Java

```

enum State {
    ON,
    OFF
}

```

Die Aufzählung **State** ergänzen wir in dem Klassendiagramm und verwenden zudem ab jetzt englische Begriffe. Die Verwendung englischer Sprache im Quellcode ist üblich, da bei den meisten Programmiersprachen für die Sprachelemente sowie die **Programmbibliotheken** ohnehin englische Sprache verwendet wird, wie auch bei Java. **Switch** und **Lamp** erhalten beide **OFF** als initialen Wert für **state**.

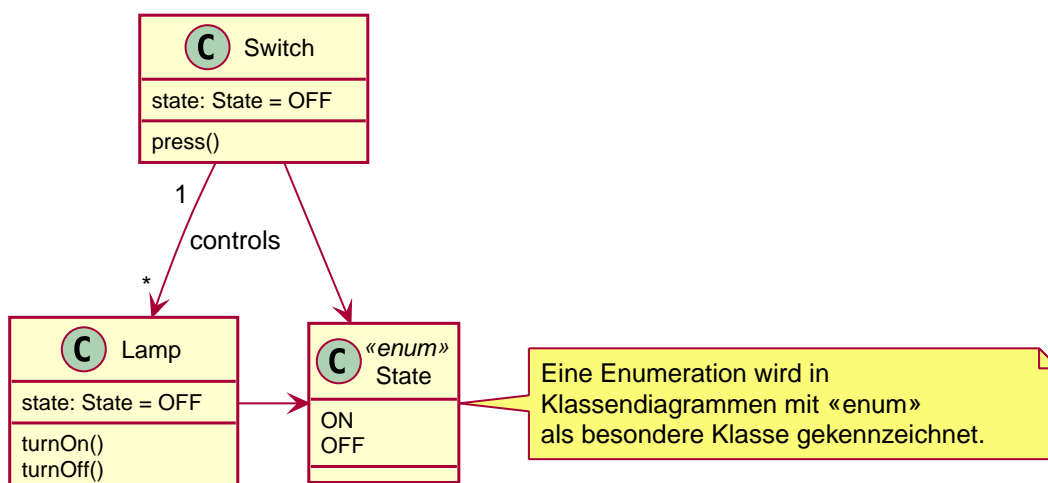


Abbildung 3. Klassendiagramm mit Enumeration und englischen Begriffen

Sowohl **Switch** als auch **Lamp** verwenden **State**, was durch Pfeile in [Abbildung 3](#) gekennzeichnet ist.

2. Objekte und Beziehungen zwischen Objekten erstellen

Um Objekte einer Klasse zu erstellen, wird ein sogenannter Konstruktor benötigt. In Java sind Konstruktoren besondere Methoden, die denselben Namen wie die Klasse tragen. Sie können Parameter haben, um z. B. Attribute zu initialisieren. In unserem Beispiel erhält **Switch** den Konstruktor **Switch()** und **Lamp** den Konstruktor **Lamp()**. Die beiden Konstruktoren haben also jeweils keinen Parameter.

Um die Beziehungen zwischen Objekten der Klasse **Lamp** und einem Objekt der Klasse **Switch** herzustellen bzw. auch wieder zu lösen, werden zusätzliche Methoden in **Switch** benötigt, für die wir mit **connect** bzw. **disconnect** möglichst passende Namen wählen. Sie haben beide einen Parameter mit dem Datentyp **Lamp**.

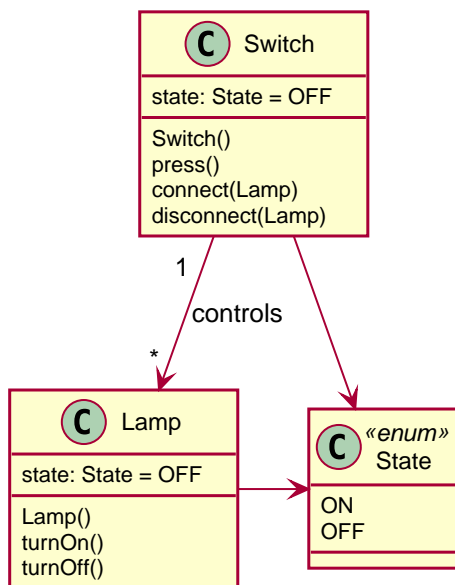


Abbildung 4. Klassendiagramm mit Enumeration und englischen Begriffen

Damit haben wir aus den User-Stories eine objektorientierte Struktur für die Software entworfen. Man nennt diesen Schritt **objektorientierte Modellierung**.

3. Die Aggregation (Hat-Beziehung)

Neben der **Assoziation (Kennt-Beziehung)** gibt es noch stärkere Beziehungstypen, u. a. die **Aggregation (Hat-Beziehung)** zwischen Objekten bzw. Klassen. So *hat* ein Motorrad ein Vorderrad.

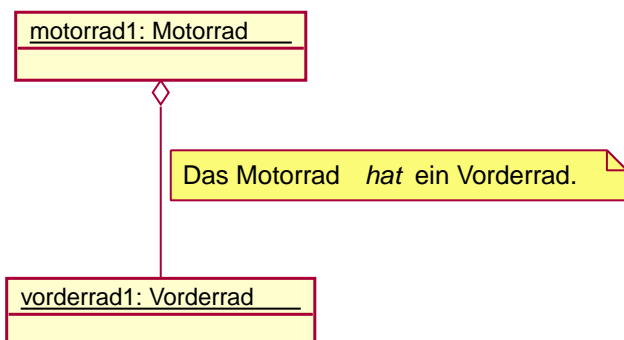


Abbildung 5. Objektdiagramm mit Aggregation

Die Raute (auch Diamand genannt), die diesen Beziehungstyp kennzeichnet, darf *nicht* wie die Pfeilspitze der Assoziation interpretiert werden, denn die Beziehung geht vom Motorrad zum Vorderrad und *nicht* vom Vorderrad zum Motorrad. Auf der Ebene der Programmierung ist bei einer Aggregation die Klasse Motorrad für die Verwaltung der Objekte der Klasse Vorderrad zuständig.