**http://ovm-kassel.de	Lernsituation**	
Lernsituation IT-AE-JA-LS-4.1 Ampelsteuerung	
Code	IT-AE-JA-LS-4.1
Autor	André Bauer <a(dot)bauer(at)ovm-kassel(dot)de>
Datum	3. Juni 2018
Links	• Vortragsfolien "Modellieren mit der Unified Modeling Language: Klassen- und Objektdiagramme" • Skript "Objektorientierte Programmierung mit Java und UML"
Verwandte Literatur	• Information IT-AE-JA-INFO-3.5 Quellcode dokumentieren
Lizenz	 Dieses Werk ist lizenziert unter einer Creative Commons Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz.

Ampelsteuerung

Die Smart Traffic Tech GmbH entwickelt u. a. Steuerungssysteme für Verkehrsampeln. Sie sollen dazu in der Entwicklungsabteilung eine Software-Komponente entwickeln.

1. Komponenten

Der für das Projekt verantwortliche Software-Architekt hat bereits einen Entwurf für die Komponenten und deren Abhängigkeiten entwickelt.

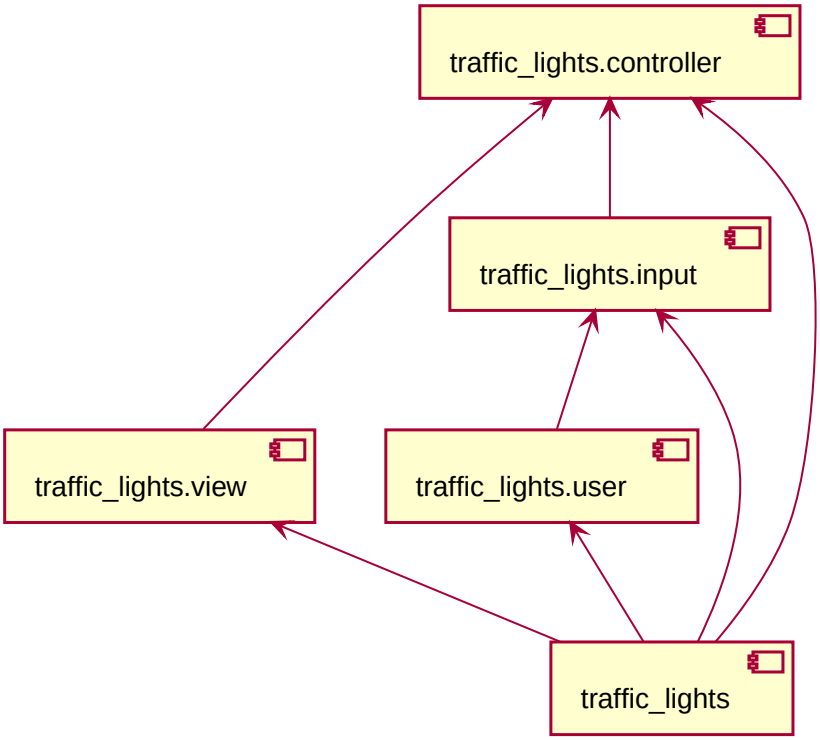

Abbildung 1. Komponenten und Abhängigkeiten

Der Projektleiter hat Ihnen und ihrem Teampartner zur Wahl gestellt, ob Sie

- die zentrale Steuerungseinheit, den Controller `com.traffic_lights.controller`,
- eine Komponente zur graphische Anzeige, den View `com.traffic_lights.view`, oder
- die Komponente für die Eingabe über eine Grafische Benutzeroberfläche, den Input `com.traffic_lights.input`,

der Ampelschaltung entwickeln.

2. Schnittstellen

Die Komponenten sollen dabei über Schnittstellen verbunden werden und nur über diese Informationen austauschen. Dies soll die Integration der Komponenten aus den verschiedenen Teams erleichtern und ermöglichen, dass Komponenten auch ausgetauscht werden können.

Die Schnittstelle der beiden Komponenten Controller und View ist durch zwei Java-Interfaces definiert, die jeweils eine interne Aufzählung der möglichen Zustände enthalten, die den Ampelphasen für eine Standard-Ampelanlage mit vier Signalisierungszuständen bzw. einer zweiphasigen Fußgängerampel entsprechen.

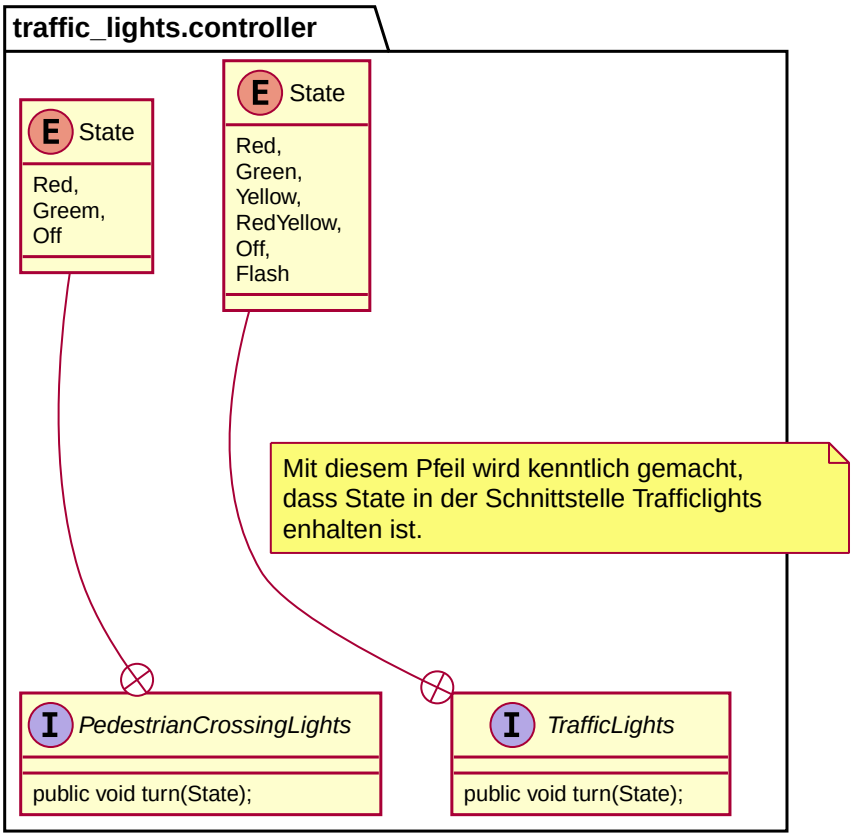

Abbildung 2. Schnittstelle zwischen den Komponenten View und Controller

Der Controller benutzt diese Schnittstelle, indem er für die Steuerungskommandos an die Ampeln mit der Methode `turn(State)` die Java-Schnittstellen PedestrianCrossingLights sowie TrafficLights anstatt konkreter Java-Klassen verwendet. In Abbildung 3 verwendet die Klasse PedestrianCrossingController die Schnittstellen zur Kommunikation mit dem View.

Der View verwendet diese Schnittstelle durch Klassen, die die Java-Schnittstellen PedestrianCrossingLights sowie TrafficLights implementieren, also indem er Klassen bereitstellt, die über eine passende Methode `turn(State)` verfügen. In Abbildung 3 implementieren die Klassen SimpleTrafficLights und SimplePedestrianCrossingLights diese Schnittstellen.

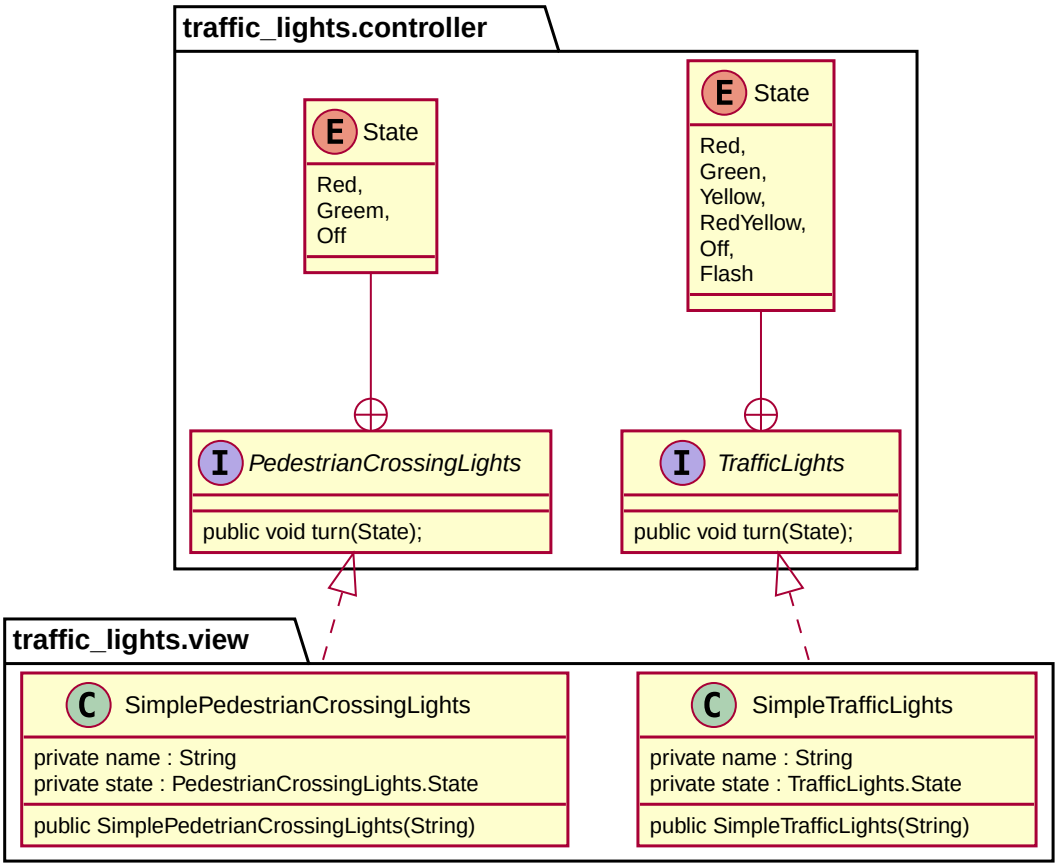

Abbildung 3. Verwendung der Schnittstelle zwischen den Komponenten View und Controller.

Die Komponenten Input und Controller kommunizieren über die Schnittstelle PedestrianInput.

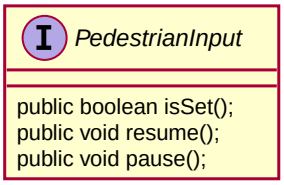

Abbildung 4. Java-Interface PedestrianInput

3. Prototyp

Ein Team hat bereits einen Prototypen als Machbarkeitsnachweis entwickelt, der als Git-Repository und als ZIP-Archiv vorliegt und dessen Aufbau in Abbildung 5 als Klassen-Diagramm dargestellt ist.

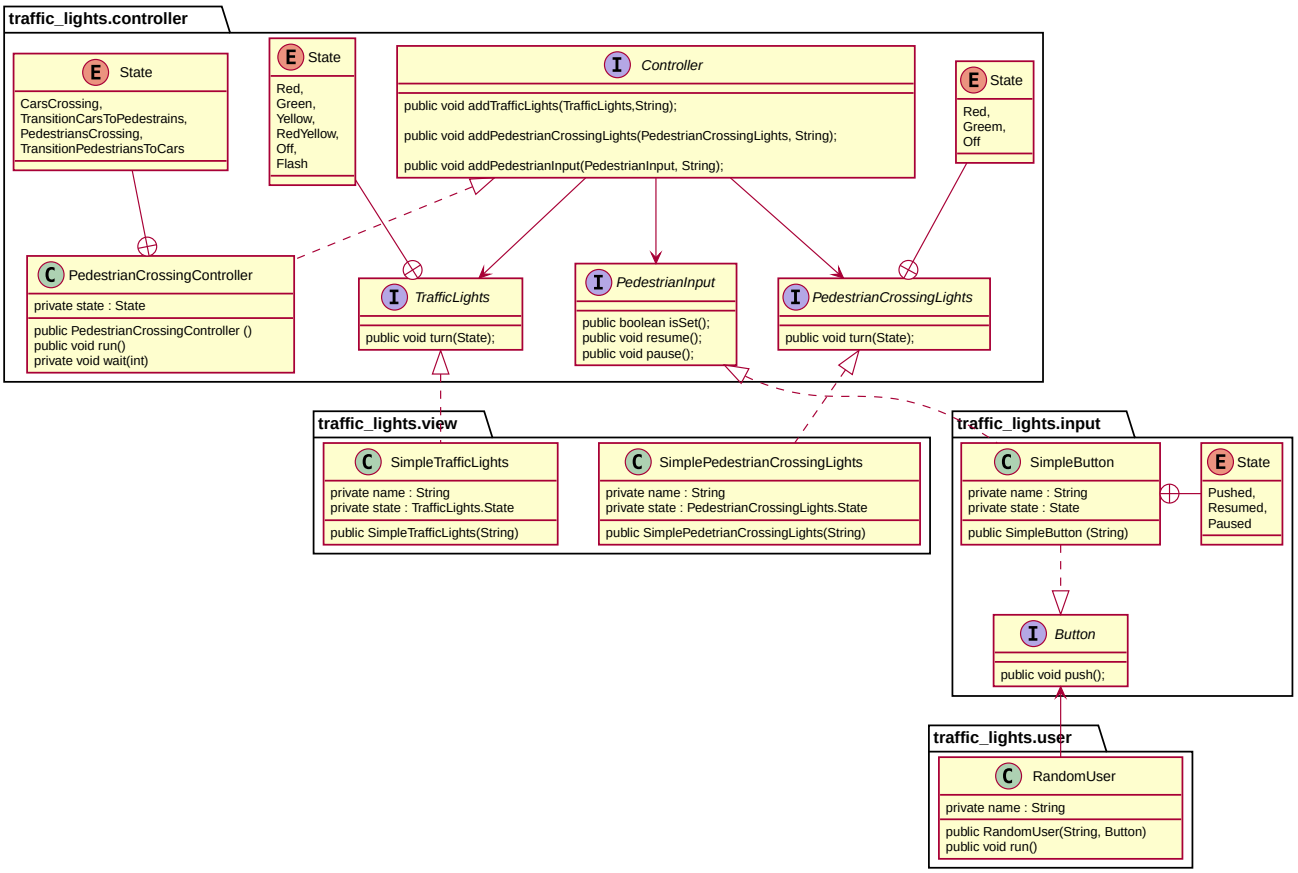

Abbildung 5. Prototyp der Ampelsteuerung

PedestrianCrossingController und RandomUser sind als Runnable implementiert, so dass diese in nebenläufigen Threads ablaufen können. Die Klasse RandomUser dient dabei als Test-Benutzer, der nach zufälligen Zeitabständen den Button über die Methode `push()` betätigt.

Eine Main-Klasse, die der Übersicht halber in Abbildung 5 *nicht* dargestellt ist, verbindet die Komponenten und startet das System.

4. Dokumentation

Die Dokumentation mit Javadoc wird in der Information IT-AE-JA-INFO-3.5 Quellcode dokumentieren näher erläutert. Da die Ampelsteuerung aus mehreren Java-Paketen besteht, wird Javadoc mit der Option `-subpackages` mitgeteilt, dass auch für alle Unterpakete die Dokumentation generiert werden soll.

```
$ javadoc -d doc -author -private -sourcepath src -subpackages com.traffic_lights
```

Die Dokumentation im Ordner `doc` kann anschließend mit einem Browser betrachtet werden.

```
$ firefox doc/index.html
```

Aufgaben

Stellen sie die Quelltexte Ihrer Lösungen als Archiv-Dateien (im `zip-` oder `tar.gz-` Format) oder als git-Repository zur Verfügung.
Diese sollten keine (`javac`-)Compiler-Fehler mehr enthalten.

1. Importieren Sie den Prototypen in Ihre Java-Entwicklungsumgebung und prüfen Sie, ob Sie diesen kompilieren und ausführen können. Die Ausgabe sollte nach etwa einer Minute ungefähr so aussehen:

```
User Random
Simple Pedestrian Crossing Lights PS1: Red
Simple Traffic Lights TS1: Red
SimpleButton B1 with State Resumed
Simple Traffic Lights TS1: Green
Simple Pedestrian Crossing Lights PS1: Red
System and User started
User Random pressed button SimpleButton B1 with State Resumed
SimpleButton B1 with State Pushed
SimpleButton B1 with State Paused
Simple Traffic Lights TS1: Yellow
User Random pressed button SimpleButton B1 with State Paused
Simple Traffic Lights TS1: Red
Simple Pedestrian Crossing Lights PS1: Green
Simple Pedestrian Crossing Lights PS1: Red
User Random pressed button SimpleButton B1 with State Paused
Simple Traffic Lights TS1: RedYellow
Simple Traffic Lights TS1: Green
SimpleButton B1 with State Resumed
User Random pressed button SimpleButton B1 with State Resumed
SimpleButton B1 with State Pushed
User Random pressed button SimpleButton B1 with State Pushed
...
```

2. Entwickeln und testen Sie mithilfe des Prototypen entweder
   a. für den View zwei Klassen `GraphicalTrafficLights` und `GraphicalPedestrianCrossingLights`, die anstatt der Text-Ausgaben von SimpleTrafficLights und SimplePedestrianCrossingLights eine graphische Ausgabe mithilfe der Java-Grafikprogrammierung anbieten,
   b. für den Input anstatt der Klasse SimpleButton eine grafische Oberfläche (GUI) oder
   c. einen Controller für die Ampfelsteuerung einer Straßenkreuzung.
   Vertiefend können Sie diesen Controller noch erweitern um
      i. einen bedarfsgesteuerten Fußgängerübergang oder
      ii. eine Fahrradampel, deren Grün-Phase zeitversetzt einige Sekunden vor den Fahrzeugen beginnt.

Das Systemverhalten können Sie zunächst mit einem Aktivitätsdiagramm planen.

3. Führen Sie einen Integrationstest durch, indem Sie Ihre Komponente mit der einer anderen Gruppe und Komponenten des Prototypen verbinden und Tests durchführen, z. B. mit JUnit.
4. Dokumentieren Sie Ihren Quellcode mit Javadoc.