Praktische Informatik II SS 2007

Übungsblatt 6

Abgabe: 11.06.2007

Aufgabe 1 Transitionssysteme am Beispiel eines Straßenbahnübergangs

Die Einsatzmöglichkeiten von Transitionssystemen sind vielfältig. In dieser Aufgabe soll eine beispielhafte Verwendung zur Kontrolle eines Straßenbahnübergangs objektorientiert realisiert werden. Dieser ist in Abbildung 1 dargestellt.

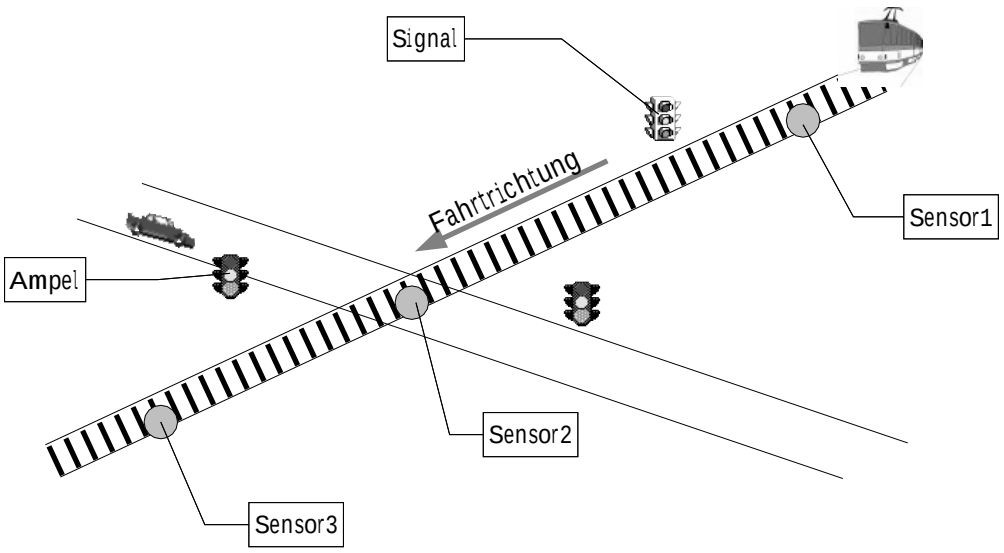

Abbildung 1: Straßenbahnkreuzung.

Dabei handelt es sich um einen sehr vereinfachten Bahnübergang. Folgende wichtige Komponenten sind dort enthalten:

- **Ampel**: Sie kennt die Ausgaben **rot** und **grün** (gleichbedeutend mit **aus**). **grün** zeigt an, dass Verkehrteilnehmer den Übergang überqueren dürfen, **rot**, dass sie es nicht dürfen.
- **Signal**: Es kennt ebenfalls die Ausgaben **rot** und **grün**. Es zeigt an, ob die Straßenbahn fahren darf (**grün**) oder nicht (**rot**).
- **Sensor1**: Dieser Sensor zeigt an, dass sich eine Straßenbahn nähert, indem er die Eingabe **Sensor1** liefert.
- **Sensor2**: Dieser Sensor zeigt an, dass eine Straßenbahn gerade den Bahnübergang überquert, indem er die Eingabe **Sensor2** liefert.
- **Sensor3**: Dieser Sensor zeigt an, dass eine Straßenbahn gerade den Bahnübergang verlässt, indem er die Eingabe **Sensor3** liefert.

Die Straßenbahnlinie ist eingleisig, passierende Straßenbahnen fahren immer in der gleichen Richtung. Folgendermaßen soll der Bahnübergang funktionieren. Sobald eine Straßenbahn den ersten Sensor passiert, soll die Ampel auf **rot** und gleichzeitig das Signal auf **grün** geschaltet werden. Der zweite Sensor dient nur zur Kontrolle und zeigt dem System, dass die Straßenbahn tatsächlich den Bahnübergang überquert. Passiert die Straßenbahn den dritten Sensor, so wird die Ampel auf **grün** und das Signal auf **rot** geschaltet. Das System ist nun wieder bereit, die nächste Straßenbahn passieren zu lassen.

Daraus folgt, dass immer die Eingaben in der Reihenfolge **Sensor1**, **Sensor2**, **Sensor3** auftreten müssen. Insbesondere ist es nicht möglich, dass zwei nachfolgende Straßenbahnen in einem Zyklus durchfahren. In dem Fall, dass eine Verletzung der Eingabefolge auftritt, soll das System einen Fehlerzustand betreten, in dem sowohl die Ampel als auch das Signal auf **rot** stehen. In diesem Zustand verharrt das System, bis ein Techniker das Problem behebt.

In Abbildung 2 ist ein Transitionssystem dargestellt, welches die oben beschriebene Funktionalität ausdrückt.

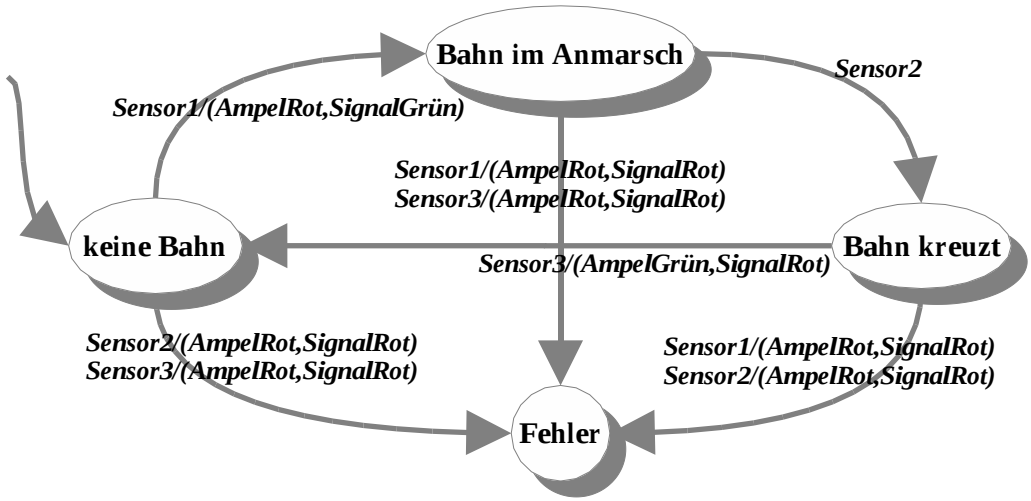

Abbildung 2: Transitionssystem für die Straßenbahnkreuzung. Mehrere Transitionen zwischen jeweils zwei Zuständen sind zu jeweils einem Pfeil zusammengefasst.

Aufgabe 1.1 Abstraktes Transitionssystem (50%)

Entwerft eine abstrakte Klasse **Transitionssystem**, die einen parameterlosen Konstruktor und folgende Methoden enthält:

public void erzeugeZustand (String zustandsName):

Fügt einen Zustand mit dem Namen **zustandsName** in das Transitionssystem ein.

public void erzeugeTransition (String quelle, Markierung markierung, String ziel):

Fügt eine Transition mit den angegebenen Quell- und Zielzuständen in das Transitionssystem ein. Die **markierung** enthält die Information, bei welcher

Eingabe die Transition schaltet und welche Ausgaben sie verursacht. Näheres zum Typ **Markierung** s.u.

public void traversiereTransitionssystem():
Diese Methode aktiviert das Transitionssystem, d.h. eine Simulation (s.u.) des Systems wird ausgeführt.

Diese Klasse soll vollständig unabhängig von der konkreten Anwendung (Straßenbahnübergang) sein.

Insbesondere soll hier noch nicht festgelegt werden, auf welche Weise die Eingaben zum Transitionssystem gelangen. Deklariert dafür eine abstrakte Methode **protected Object holeEingabe()**, die von abgeleiteten Klassen definiert wird und es so ermöglicht, konkrete Eingaben auf unterschiedliche Weise zu ermitteln.

Implementiert die Klasse **Transitionssystem** so, dass sie selbst eine Liste von Zuständen enthält. Eine Transition wiederum wird dann jeweils in dem Zustand verwaltet, der der Quellzustand dieser Transition ist. Beachtet auch, dass einer der Zustände der Anfangszustand sein muß, damit die Simulation (s.u.) einen definierten Anfang hat. Seht dafür eine weitere Methode vor:

public void setzeAnfangszustand (String zustandsName):
Macht den Zustand mit dem Namen **zustandsName** zum Anfangszustand.

Beachtet bei der Implementierung der Methoden, dass möglicherweise ungültige Werte übergeben werden können, wie z. B. ein Quellzustand, der im Transitionssystem gar nicht existiert. Lasst Eure Methoden dabei angemessen reagieren. Beachtet vergleichbare Probleme auch in den anderen Klassen.

Entwerft eine Klasse **Zustand**, die die Zustände des Transitionssystems repräsentiert. Insbesondere soll jedes Objekt dieser Klasse Referenzen auf genau die Transitionen enthalten, deren Quellzustand es ist. Ferner hat jeder Zustand einen Namen, der ihn identifiziert; folglich sollen sich zwei Zustände gleichen, wenn sie den gleichen Namen haben. Dafür ist es wie immer hilfreich, die automatisch von Object geerbte Methode **public boolean equals (Object o)** zu redefinieren.

Entwerft eine Klasse **Transition**, welche die Transitionen des Transitionssystems repräsentiert. Hier soll die Referenz auf den Zielzustand gespeichert werden. Weiterhin enthält sie Markierungen (Labels).

Entwerft dafür eine abstrakte Klasse oder Schnittstelle **Markierung**, welche den folgenden Zweck hat:

Jede Transition enthält eine Markierung. In der Markierung wiederum ist festgelegt, welche Eingabe die zugehörige Transition auslösen soll. Weiterhin ist in der Markierung enthalten, welche Ausgaben erzeugt werden, wenn die zugehörige Transition ausgelöst wird. Damit nun sowohl **Zustand** als auch **Transition** (wie auch **Transitionssystem**) anwendungsunabhängig bleiben, müßt Ihr in der Klasse **Markierung** zwei Methodendeklarationen vorsehen: Der Zweck der ersten ist, zu prüfen, ob ein gegebenes Ereignis mit dem in der Markierung übereinstimmt. Die zweite wird benötigt, um die Ausgaben zu erzeugen. Dann könnt Ihr **Markierung** verwenden, um es in **Transition** zu referenzieren.

Bis hierhin habt ihr eine allgemein verwendbare Klassenstruktur für Transitionssysteme geschaffen, in der die Simulation folgendermaßen ausgeführt werden kann:

1. In einer Endlosschleife wird jeweils eine Eingabe ermittelt.
2. Diese Eingabe wird dem aktuellen Zustand übergeben.
3. Der Zustand bietet allen seinen ausgehenden Transitionen diese Eingabe an, bis eine Transition durch diese Eingabe ausgelöst wird oder alle sie abgelehnt haben.
4. Jede Transition überlässt es dabei ihrer Markierung zu entscheiden, ob sie ausgelöst wird. Falls ja, lässt sie die Markierung auch die Ausgaben erzeugen und gibt ihrem Quellzustand (welches der aktuelle Zustand ist) den Folgezustand zurück.
5. Der aktuelle Zustand gibt nun diesen Zustand an das Transitionssystem zurück. Wurde keine Transition ausgelöst, so gibt er sich selbst zurück.
6. Somit hat das Transitionssystem nun einen (neuen) aktuellen Zustand und setzt die Iteration fort.

Aufgabe 1.2 Die konkrete Tramsteuerung (50%)

Nun implementiert konkrete Klassen, welche auf die Anwendung zugeschnitten sind. Das sind:

- Eine Klasse **Tramsteuerung**, die von **Transitionssystem** abgeleitet ist. Sie soll das Ermitteln der Eingaben übernehmen. Eingaben sollen dabei die Zeichenketten „Sensor1“, „Sensor2“ und „Sensor3“ sein. Sie sollen als interaktive Konsoleneingabe eingelesen werden (dafür könnt Ihr die Klasse **Input** verwenden, die Ihr in auf die PI2 Webseite findet). Andere Zeichenketten sollen zurückgewiesen werden.
- Erstellt zu jeder Markierung, die Ihr benötigt (also „Sensor1/(AmpelRot,SignalGrün)“ etc.), eine Klasse, welche von **Markierung** abgeleitet ist bzw. sie implementiert. Implementiert in der jeweiligen Klasse das Prüfen der entsprechenden Eingabe sowie das Erzeugen der jeweiligen Ausgaben. Dabei ist es hinreichend, die Ausgaben in Form von Textausgaben auf die Konsole zu realisieren.

Nun soll in der Hauptroutine des Programms eine **Tramsteuerung** erstellt werden. Dieser wird dort zunächst mit Zuständen und Transitionen gefüllt, dann wird die Simulation gestartet:

```
Tramsteuerung tramsteuerung = new Tramsteuerung();
tramsteuerung.erzeugeZustand("keine Bahn");
tramsteuerung.erzeugeZustand("Bahn im Anmarsch");
tramsteuerung.erzeugeZustand("Bahn kreuzt");
tramsteuerung.erzeugeZustand("Fehler");
tramsteuerung.setzeAnfangszustand("keine Bahn");
tramsteuerung.erzeugeTransition("keine Bahn",
                                new Markierung(Sensor1AmpelRotSignalGruen),
                                "Bahn im Anmarsch");
...
tramsteuerung.traversiereTransitionssystem();
```