Vorlesung: Dr.-Ing. Matthias Becker - xmb@sim.uni-hannover.de
Übung: Philipp Schiffmann - schiffmann@sim.uni-hannover.de
Florian Chudigiewitsch - fch@thi.uni-hannover.de

Assignment 4

Interfaces, Strings, jar

Abgabe: **16.05.2018, 13:45 Uhr**

I. Statistische Auswertung

Ziel einer Simulation ist immer, etwas über das modellierte System zu erfahren, etwa zur Optimierung einer Kenngröße. Die Verkehrssimulation, die wir gerade entwickeln, könnte zum Beispiel einer Verkehrsbehörde ermöglichen, ihr Straßennetz auf minimale durchschnittliche Wartezeit für Autos zu optimieren.

Dazu muss das Programm die Möglichkeit haben, Kenngrößen zu protokollieren. Wir interessieren uns dabei für folgende Werte, die in jedem Zeitschritt für alle Objekte aufgezeichnet werden sollen: Für Autos die im letzten Zeitschritt zurückgelegte Strecke (entweder 0 oder 1); für Straßen die aktuelle Anzahl an darauf stehenden Autos (zwischen 0 und Länge der Straße); und für Ampelkreuzungen die Anzahl an Autos, die gerade auf ihre Grünphase warten (zwischen 0 und summierter Länge aller ankommenden, roten Fahrbahnen).

Um deinen Code ubersichtlich zu halten, sollst du wieder Vererbung benutzen, um doppelten Code zu vermeiden. Diese drei Klassen haben bisher keinen gemeinsamen Obertyp, und die Klasse `Actor` eignet sich nicht, weil die Fahrbahn keine `update`-Methode benotigt. Definiere deshalb ein neues Interface DataSource im subpackage profiling und implementiere es in den oben genannten Klassen. Dieses Interface enthält die Methoden `void logStatus()` und `double[] getLog()`, die fur das jeweilige Objekt den aktuellen Wert in ein internes Protokoll-Array schreiben bzw. dieses herausgeben.

`logStatus` soll fur alle `DataSources` in jedem Zeitschritt aufgerufen werden, allerdings erst, nachdem sämtliche Actors aktualisiert wurden. Lege außerdem deine `Statistics`-Klasse aus assignment 1 in das `profiling` package.

II. Kommandozeilenparameter

Eine weitere Anforderung an ein „echtes“ Programm ist, dass es Benutzereingaben verarbeiten kann. Deine Simulation soll diese Eingaben als Argumente auf der Kommandozeile entgegennehmen und die folgenden Eingaben beachten. Alle Argumente sind optional.

1. `-help` gibt einen Hilfetext zur Benutzung des Programms aus und terminiert, ohne die Simulation auszuführen. Dies soll ebenfalls passieren, wenn der Benutzer ein nicht unterstütztes Argument eingibt.
2. `-seed <value>` setzt den seed des Zufallszahlengenerators auf den angegebenen Wert.
3. `-duration <value>` bestimmt die Anzahl von ausgeführten Zeitschritten in der Simulation.

4. `-inspect <name1>,<name2>,...` lässt den Status der `DataSources` mit den angegebenen Namen in jedem Zeitschritt ausgeben. `Actors` sollen ihren Status standardmäßig nicht mehr ausgeben. Außerdem sollen für die genannten Objekte am Ende der Simulation der Durchschnittswert und die Standardabweichung des protokollierten Werts ausgegeben werden.

5. `-profile-cars` sorgt dafür, dass nach dem Ende der Simulation die durchschnittliche Reisezeit über alle Autos, die das System durchlaufen haben, berechnet und ausgegeben wird.

III. JAR

Um die Benutzung des Programms zu vereinfachen, sollst du das fertige Programm in ein .jar verpacken.

IV. Lösungshinweise

1. Kommandozeilenparameter übergibst du beim Programmstart direkt hinter dem Namen der `Main`-Klasse, z.B.: `java Main -help` oder `java Main -seed 4 -inspect A,B,C,A->C,B->C -duration 100`

2. Die Werte dieser Parameter stehen in `args` (oder wie auch immer du die Variable genannt hast) der Methode `static void main(String[] args)`. Die Parameter werden an Leerzeichen getrennt und landen jeweils in eigenen Array-Indices.

3. Eine `jar`-Datei erstellst du mit dem jar-Befehl. Anschließend führst du sie mit dem Befehl `java -jar <Dateiname>` aus. Denk daran, eine Manifest-Datei hinzuzufügen, damit Java beim Ausführen die Main-Datei automatisch findet.

V. Bewertungskriterien

1. Die Klassen Auto, Fahrbahn und Kreuzung implementieren das `DataSource` Interface.

2. Dein Programm akzeptiert Eingaben über die Kommandozeile.

3. Das funktionierende Programm liegt als ausführbare `.jar`-Datei im repository.