****

**FH - Studiengang für**

**Informationstechnik und System-Management**

**Salzburg**

**ITS**

**Übungen in**

**Spezielle Softwaretechnologien**

**Protokoll**

Gegenstand der Übung gemäß Anleitung:

**Verkehrssimulation**

**Version: 1**

**Datum der Übung: 01.03.2017**

**Datum der Abgabe: 13.07.2017**

**Autoren: Christopher Wieland, Martin Wieser, Stephanie Kaschnitz, Hannes Kleiner, Andreas Lippmann**

Inhaltsverzeichnis

[1 Aufgabenstellung 1](#_Toc487535096)

[2 Gesamtübersicht und Idee 2](#_Toc487535097)

[2.1 GUI 2](#_Toc487535098)

[2.1.1 Klasse „1“ 2](#_Toc487535099)

[2.1.2 Klasse „2“ 2](#_Toc487535100)

[2.1.3 Klasse „3“ 2](#_Toc487535101)

[2.2 Verkehrsnetz 2](#_Toc487535102)

[2.2.1 Klasse „1“ 2](#_Toc487535103)

[2.2.2 Klasse „2“ 3](#_Toc487535104)

[2.3 Verkehrsregeln 3](#_Toc487535105)

[2.3.1 Klasse „1“ 3](#_Toc487535106)

[2.3.2 Klasse „2“ 3](#_Toc487535107)

[2.3.3 Klasse „2“ 3](#_Toc487535108)

[2.4 Verkehrsteilnehmer 3](#_Toc487535109)

[2.4.1 Klasse „1“ 3](#_Toc487535110)

[2.4.2 Klasse „2“ 3](#_Toc487535111)

[2.4.3 Klasse „2“ 3](#_Toc487535112)

[2.5 Ampelsteuerung 4](#_Toc487535113)

[2.5.1 Klasse „Ampeln“ 4](#_Toc487535114)

[2.5.2 Klasse „Ampelsteuerung“ 4](#_Toc487535115)

[2.6 RabbitMQ 4](#_Toc487535116)

[2.6.1 Klasse „1“ 4](#_Toc487535117)

[2.6.2 Klasse „2“ 4](#_Toc487535118)

[2.6.3 Klasse „2“ 4](#_Toc487535119)

[3 Dokumentation der Funktionalität der Verkehrssimulation 5](#_Toc487535120)

[3.1 GUI 5](#_Toc487535121)

[3.1.1 Funktionen 5](#_Toc487535122)

[3.1.2 Deklarierung der Funktionen 5](#_Toc487535123)

[3.2 Verkehrsnetz 6](#_Toc487535124)

[3.2.1 Funktionen 6](#_Toc487535125)

[3.2.2 Deklarierung der Funktionen 6](#_Toc487535126)

[3.3 Verkehrsregeln 7](#_Toc487535127)

[3.3.1 Funktionen 7](#_Toc487535128)

[3.3.2 Deklarierung der Funktionen 7](#_Toc487535129)

[3.4 Verkehrsteilnehmer 8](#_Toc487535130)

[3.4.1 Funktionen 8](#_Toc487535131)

[3.4.2 Deklarierung der Funktionen 8](#_Toc487535132)

[3.5 Ampelsteuerung 9](#_Toc487535133)

[3.5.1 Funktionen der Klasse Ampel 9](#_Toc487535134)

[3.5.2 Funktionen der Klasse Ampelsteuerung 10](#_Toc487535135)

[3.5.3 Variablen der Klasse Ampelsteuerung 12](#_Toc487535136)

[3.6 RabbitMQ 12](#_Toc487535137)

[3.6.1 Funktionen 12](#_Toc487535138)

[3.6.2 Deklarierung der Funktionen 12](#_Toc487535139)

[4 Zusätzliche externe Komponenten 13](#_Toc487535140)

[5 Zusammenfassung und Ausblick 14](#_Toc487535141)

[5.1 Derzeit nicht implementiert 14](#_Toc487535142)

[5.2 Ausblick 14](#_Toc487535143)

Tabellenverzeichnis

[Tabelle 3.1: Funktionen der GUI 5](#_Toc487535144)

[Tabelle 3.1: Funktionen des Verkehrsnetzes 6](#_Toc487535145)

[Tabelle 3.1: Funktionen der Verkehrsregeln 7](#_Toc487535146)

[Tabelle 3.1: Funktionen der Verkehrsteilnehmer 8](#_Toc487535147)

[Tabelle 3.1: Funktionen der Getter und Setter - Ampelsteuerung 9](#_Toc487535148)

[Tabelle 3.1: Funktionen des RabbitMQ 12](#_Toc487535149)

# Aufgabenstellung

# Gesamtübersicht und Idee

Für die Umsetzung der Aufgabenstellung wurden die Aufgaben in sechs große Bereiche gespalten: GUI, Verkehrsnetz, Verkehrsteilnehmer, Verkehrsregeln, Ampelsteuerung und Rabbit MQ. Zu beachten ist der Punkt, dass die Ampelsteuerung extern zur Verfügung gestellt werden musste. Um eine Kommunikation zwischen Ampelsteuerung und Verkehrssimulation zu erreichen, wurde mit Windows Communication Foundation (WCF) gearbeitet. Hierbei ist die Ampelsteuerung der Server, welcher den Dienst der Ampel zur Verfügung stellt und die Verkehrssimulation der Client, welcher den Status der Ampeln abgefragt.

## GUI

Beschreibung

### Klasse „1“

Beschreibung

### Klasse „2“

Beschreibung

### Klasse „3“

Beschreibung

## Verkehrsnetz

Beschreibung

### Klasse „1“

Beschreibung

### Klasse „2“

Beschreibung

## Verkehrsregeln

Beschreibung

### Klasse „1“

Beschreibung

### Klasse „2“

Beschreibung

### Klasse „2“

Beschreibung

## Verkehrsteilnehmer

Beschreibung

### Klasse „1“

Beschreibung

### Klasse „2“

Beschreibung

### Klasse „2“

Beschreibung

## Ampelsteuerung

Die Ampelsteuerungskomponente hat die Aufgabe, Ampeln nach Belieben zu erstellen, aktiv/inaktiv zu schalten, Statusphasen manuell anzupassen und automatisch umzuschalten, wenn die Phasen jeweils vorüber sind. Realisiert wurde diese Komponente mit 2 Klassen:

### Klasse „Ampeln“

Diese Klasse symbolisiert eine einzelne Ampel. Mithilfe dieser Klasse ist das Erstellen von Ampeln ermöglicht worden. Funktionen und Variablen zu dieser Klasse sind in Abschnitt 3.5.1 ersichtlich.

### Klasse „Ampelsteuerung“

Die Klasse Ampelsteuerung ist für das Verhalten der Ampeln verantwortlich. Hier wird ebenso der WCF Server gestartet. In Kapitel 3.5.2 werden die verwendeten Funktionen und Variablen beschrieben.

## RabbitMQ

Beschreibung

### Klasse „1“

Beschreibung

### Klasse „2“

Beschreibung

### Klasse „2“

Beschreibung

# Dokumentation der Funktionalität der Verkehrssimulation

In diesem Sektor werden alle Funktionen aller Komponenten erklärt. Ebenso wird darauf eingegangen, wie alle Komponenten miteinander Funktionieren.

## GUI

Beschreibung

### Funktionen

Beschreibung

* Beispielfunktion 1

Beschreibung der Funktion

* Beispielfunktion 2

Beschreibung der Funktion

* Beispielfunktion 3

Beschreibung der Funktion

### Deklarierung der Funktionen

Beschreibung

|  |  |
| --- | --- |
| Funktion | Erklärung |
| Funktion 1 | Beschreibung 1 |

Tabelle .: Funktionen der GUI

## Verkehrsnetz

Beschreibung

### Funktionen

Beschreibung

* Beispielfunktion 1

Beschreibung der Funktion

* Beispielfunktion 2

Beschreibung der Funktion

* Beispielfunktion 3

Beschreibung der Funktion

### Deklarierung der Funktionen

Beschreibung

|  |  |
| --- | --- |
| Funktion | Erklärung |
| Funktion 1 | Beschreibung 1 |

Tabelle .: Funktionen des Verkehrsnetzes

## Verkehrsregeln

Beschreibung

### Funktionen

Beschreibung

* Beispielfunktion 1

Beschreibung der Funktion

* Beispielfunktion 2

Beschreibung der Funktion

* Beispielfunktion 3

Beschreibung der Funktion

### Deklarierung der Funktionen

Beschreibung

|  |  |
| --- | --- |
| Funktion | Erklärung |
| Funktion 1 | Beschreibung 1 |

Tabelle .: Funktionen der Verkehrsregeln

## Verkehrsteilnehmer

Beschreibung

### Funktionen

Beschreibung

* Beispielfunktion 1

Beschreibung der Funktion

* Beispielfunktion 2

Beschreibung der Funktion

* Beispielfunktion 3

Beschreibung der Funktion

### Deklarierung der Funktionen

Beschreibung

|  |  |
| --- | --- |
| Funktion | Erklärung |
| Funktion 1 | Beschreibung 1 |

Tabelle .: Funktionen der Verkehrsteilnehmer

## Ampelsteuerung

Die Ampelsteuerung besteht aus 2 Klassen, siehe Kapitel 2.5.1 und 2.5.2 sowie deren Methoden und Variablen. In diesem Abschnitt wird genauer auf diese eingegangen.

### Funktionen der Klasse Ampel

Damit die Phasendauer einer Ampel, der Ausfall einer Ampel oder die ID abgefragt werden kann, wurden Getter und Setter sowie variablen in der Klasse angelegt:

|  |  |
| --- | --- |
| Funktion | Erklärung |
| getStatus() | Holt Ampelstatus |
| getRotPhase() | Gibt Zeit der Rotphase wieder |
| getGelbPhase() | Gibt Zeit der Gelbphase wieder |
| getGruenPhase() | Gibt Zeit der Grünphase wieder |
| getDefect() | Gibt Defekt der Ampel an |
| getID() | Holt ID der Ampel |
| getSekundenzähler() | Erhält Sekundenzähler |
| setSekundenzähler(int value) | Setzt den Sekundenzähler |
| setRotPhase(int value) | Setzt die Rotphasendauer |
| setGelbPhase(int value) | Setzt die Gelbphasendauer |
| setGruenPhase(int value) | Setzt die Grünphasendauer |
| setDefect(int value) | Setzt eine Ampel Defekt |
| setStatus(int value) | Setzt den Status einer Ampel |
| setID(int value) | Setzt bei Erstellung die ID der Ampel |

Tabelle .: Funktionen der Getter und Setter - Ampelsteuerung

Die Variablen hierzu sind:

int sekundenzähler = 0;  
 int Status;

int ID;

bool defect = false;

int rotphase = 3;

int gelbphase = 1;

int gruenphase = 3;

### Funktionen der Klasse Ampelsteuerung

Damit die Ampelsteuerung für die Verkehrssimulation verfügbar ist und per WCF erreichbar ist, muss folgendes hinzugefügt werden:

[ServiceBehavior(Name = "Ampelsteuerung", InstanceContextMode = InstanceContextMode.Single)]

Hiermit gibt man die Ampelsteuerung unter dem Namen „Ampelsteuerung“ am Localhost bekannt und lässt somit den Zugriff auf diese zu. Weiters werden folgende Funktionen benötigt:

* private void StartServer();

Diese Funktion erstellt einen ServiceHost, welcher von der Verkehrssimulation zum Erstellen einer Verbindung benötigt wird. Weiters wird gewartet, bis von der Verkehrssimulation die Anzahl der benötigten Ampeln geliefert wird. Die Ampeln werden erstellt und ein Timer zum Schalten der Phasen wird pro Ampel gestartet.

* private Task HandleTimer();

Dieser Timer inkrementiert den Sekundenzähler aus der Klasse „Ampeln“ und erzeugt so einen Phasenwechsel.

* public int getAmpelStatus(int ampelid);

Diese Funktion erhält die eine AmpelID. Diese ID zeigt auf eine Ampel und gibt den Status dieser Ampel zurück. Wird „0“ anstelle der AmpelID mitgegeben, so erhält man den Status aller erstellten Ampeln.

* public string getAmpelAusfall(int ampelid)

Diese Funktion gibt Information darüber, ob eine Ampel ausgefallen ist aufgrund der mitgegebenen AmpelID.

* public void setAmpelAusfall(int ampelid)

Mit setAmpelAusfall ist man in der Lage, eine Ampel aufgrund dessen ID Auszuschalten.

* public void setAmpelOn(int ampelid);

Hier kann man eine Ampel mit mitgegebener ID wieder einschalten, sofern sie ausgeschaltet ist.

* public void setAmpelStatus(int ampelid, int neuerStatus);

Hier kann man den Status einer Ampel manipulieren bzw. angeben, ob eine Ampel Rot, Gelb, oder Grün ist.

* public string getRotPhase(int ampelid);

Hier erhält man aufgrund der ID die Zeit der Rotphase wieder.

* public string getGelbPhase(int ampelid);

Hier erhält man aufgrund der ID die Zeit der Gelbphase wieder.

* public string getGruenPhase(int ampelid);

Hier erhält man aufgrund der ID die Zeit der Grünphase wieder.

* public void setRotPhase(int ampelid, int zeit);

Hier kann man aufgrund der ID die Zeit der Rotphase einstellen.

* public void setGelbPhase(int ampelid, int zeit);

Hier kann man aufgrund der ID die Zeit der Gelbphase einstellen.

* public void setGruenPhase(int ampelid, int zeit);

Aufgrund der ID der Ampel wird hier die Zeit der Grünphase eingestellt.

* public void setAmpelAnzahl(int anzahl);

Diese Funktion wird zu Beginn ausgeführt, um die Anzahl der Ampeln zu erstellen. Diese Funktion wird von der Verkehrssimulation verwendet.

* public List<Ampeln> factory(int anzahl);

Diese Funktion erstellt die Ampeln.

* static void Main();

In der Main wird die Ampelsteuerung ausgeführt.

Alle Funktionen werden über das Interface „IAmpelService“ der Verkehrssimulation zur Verfügung gestellt.

### Variablen der Klasse Ampelsteuerung

Damit die Funktionen miteinander Funktionieren, wurden folgende Variablen benötigt:

ServiceHost host;

bool \_serverRunning = false;

bool run = false;

public static List<Ampeln> Trafficlights = new List<Ampeln>();

public static int Anzahl = 0;

static Timer Ampeltimer;

## RabbitMQ

Beschreibung

### Funktionen

Beschreibung

* Beispielfunktion 1

Beschreibung der Funktion

* Beispielfunktion 2

Beschreibung der Funktion

* Beispielfunktion 3

Beschreibung der Funktion

### Deklarierung der Funktionen

Beschreibung

|  |  |
| --- | --- |
| Funktion | Erklärung |
| Funktion 1 | Beschreibung 1 |

Tabelle .: Funktionen des RabbitMQ

# Zusätzliche externe Komponenten

Welche externen Komponenten kamen zum Einsatz?

# Zusammenfassung und Ausblick

Was wurde geschafft und was nicht, bzw. wo gibt’s noch kleine Probleme

## Derzeit nicht implementiert

## Ausblick