



Technische Universität Berlin

Software and Embedded Systems Engineering Group

Prof. Dr. Sabine Glesner

www.sese.tu-berlin.de    Sekr. TEL 12-4    Ernst-Reuter-Platz 7    10587 Berlin




# Softwaretechnik und Programmierparadigmen WiSe 2023/2024

Prof. Dr. Sabine Glesner  
Simon Schwan  
Julian Klein

## Übungsblatt 14

Beispielösung

### Hinweis: USE-OCL



Ihr könnt eure OCL Ausdrücke und Bedingungen mit USE-OCL der Uni Bremen testen.  USE-OCL erlaubt zwar die graphische Darstellung von Klassen- und Objektdiagrammen, die Klassendiagramme müssen jedoch textuell im .use-Format definiert werden. Objekte werden mit Hilfe von Kommandozeilen-Befehlen angelegt. Um Befehle zum Anlegen und verlinken von Objekten wiederholt ausführen zu können, können Objekt-Skripte verwendet werden. Für diese Übung stehen euch auf ISIS ein vordefiniertes Klassendiagramm (`autowerkstatt.use`) sowie ein Objekt-Skript (`autowerkstatt.soil`), das ein Beispiel-Objektdiagramm erstellt, zur Verfügung. Dort findet ihr außerdem die Datei `autowerkstatt.default.clt`, mit der ein Default-Layout für das Klassendiagramm geladen wird. Modelldateien (`.use`) und die Skripte (`.soil`) können mit einem beliebigen Editor geöffnet und bearbeitet werden. Für das Bearbeiten von Modellen bietet die GUI von USE keine Möglichkeit. Weiteres im Video USE-OCL-Tool-Einführung.

Um USE-OCL zu verwenden müsst ihr es nur herunterladen, entpacken und dann je nach Betriebssystem eins der Skripte `bin/use` oder `bin/start.use.bat` ausführen. Es öffnen sich dann sowohl eine GUI als auch die Kommandozeile zur Eingabe von Skript-Befehlen. Das Klassendiagramm könnt ihr in der GUI über den Dialog 'File → Open specification' laden, zum Laden des Objekt-Skripts müsst ihr in der Kommandozeile folgendes eingeben:

```
use> open autowerkstatt.soil
```

---

### Schlüssel:

-  Ein ergänzendes Video wird zur Vor- oder Nachbereitung veröffentlicht.
-  Wird im Tutorium besprochen.

In der GUI könnt ihr über den Button “OCL” beliebige OCL-Bedingungen auswerten. Verwendet die in ISIS zur Verfügung gestellten Ressourcen zur Erstellung des Modells.

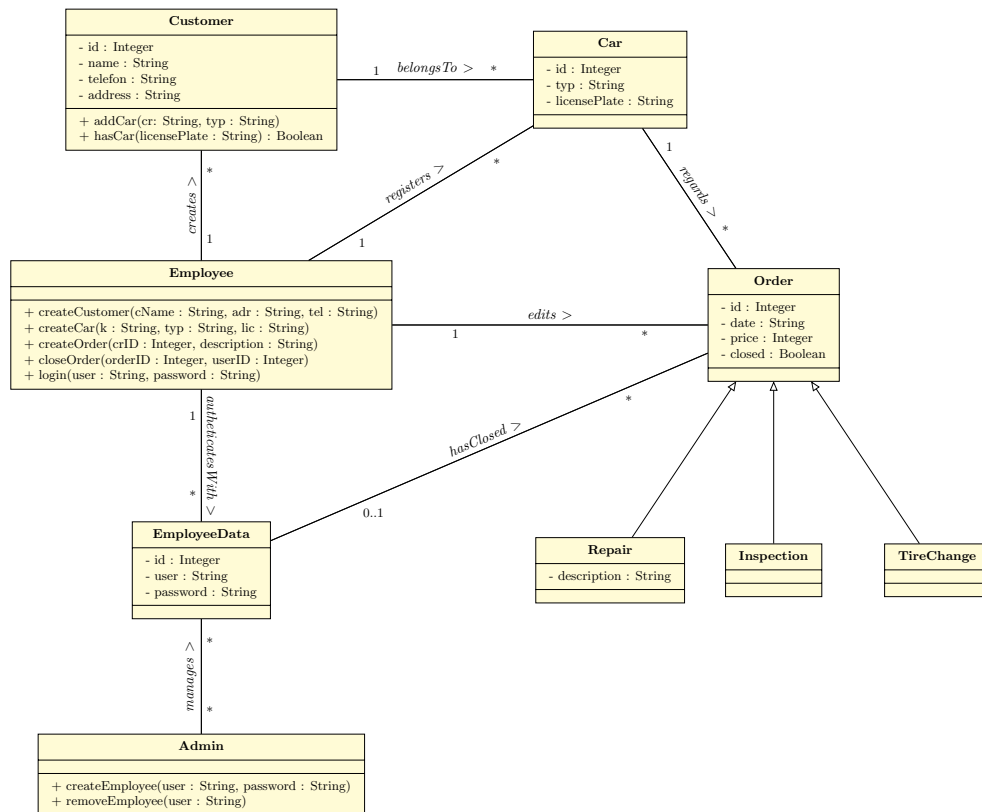


Abbildung 1: Klassendiagramm der Autowerkstatt

## Aufgabe 1: Objektdiagramme









Öffnet das Klassendiagramm und den Systemzustand aus den Vorgaben. In der graphischen Oberfläche könnt Ihr das Klassendiagramm und das Objektdiagramm betrachten (Hinweis: zur besseren Übersicht können Controller-Klassen ausgeblendet werden).

- Wodurch unterscheidet sich das Objektdiagramm vom Klassendiagramm? Warum gibt es keine Multiplizitäten?
- Passt der Systemzustand zur Spezifikation? Falls nicht, nehmt entsprechend Änderungen vor.

**Lösung:** Siehe Abbildung 2. In der Vorgabe fehlten die Verbindungen i2,f2 und f1,c1.

## Aufgabe 2: OCL Werte

Verwendet OCL um folgende Werte des Systemzustands zu erhalten.








- a) Die Id des Kunden/der Kundin **c1** 
- b) Das Kennzeichen des Fahrzeugs **f1**  
- c) Der Name des Besitzers/der Besitzerin des Fahrzeugs **f1** 
- d) Der Mitarbeiter/die Mitarbeiterin, der/die den Auftrag **r2** beendet hat 
- e) Diejenige Reparatur aus **re3** und **re4** mit dem höheren Preis 
- f) Den ersten Buchstaben des Kennzeichens von Fahrzeug **f1**  

### Lösung:

- a) `c1.id`
- b) `f1.licensePlate`
- c) `f1.customer.name`
- d) `r2.closing` (siehe Rollenbezeichner / Assoziationsenden im Klassendiagramm)
- e) `if re3.price > re4.price then re3 else re4 endif`
- f) `f1.licensePlate.substring(1,1)`

### Aufgabe 3: OCL Bedingungen

Logische Ausdrücke sind statisch verifizierbar und werden auf verschiedene Weise eingesetzt. Formalisiert die folgenden logischen Aussagen für das gegebene Klassenmodell.













- a) Sind die Namen der Kund:innen `c1` und `c2` gleich? 
- b) Gehört das Fahrzeug `f1` dem Kunden/der Kundin `c1`?  
- c) Hat das Kennzeichen des Fahrzeugs `f1` einen nicht-leeren Wert? 
- d) Ist der Auftrag `i1` eine Inspektion? 
- e) Ist das Objekt `i1` ein Auftrag?  

#### Lösung:

- a) `c1.name = c2.name`
- b) `f1.customer = c1` (hier noch auf Mengen verzichten)
- c) `f1.licensePlate <> ''` oder `f1.licensePlate.size() > 0`  
Hier auch möglich Undefined (null) zu berücksichtigen:  
`f1.licensePlate <> Undefined and f1.licensePlate <> ''` oder  
`f1.licensePlate <> null and f1.licensePlate <> ''`
- d) `i1.oclIsTypeOf(Inspection)`
- e) `i1.oclIsKindOf(Order)`

## Aufgabe 4: OCL Collections

Definiert die folgenden Mengen:









- a) Alle Fahrzeuge im System 
- b) Die Fahrzeuge von `c1` 
- c) Die Fahrzeuge von `c1` und das Fahrzeug `f4` 
- d) Alle Fahrzeuge von `c1` und `c2` 
- e) Die Fahrzeuge die gleichzeitig `c1` und `c2` gehören.  
- f) Die Typen aller Fahrzeuge des Customers `c1`. 
- g) Die Preise aller Aufträge im System.  
- h) Die Anzahl der Aufträge für das Fahrzeug `f4`. 
- i) Die Anzahl der Kund:innen mit der ID 12  

### Lösung:

- a) `Car.allInstances()`
- b) `c1.car`
- c) `c1.car->including(f4)`
- d) `c1.car->union(c2.car)`
- e) `c1.car->intersection(c2.car)`
- f) `c1.car.typ`
- g) `Auftrag.allInstances().price`
- h) `f4.order->size()`
- i) `Customer.allInstances.id->count(12)`

## Aufgabe 5: OCL Aussagen

Überprüft mithilfe von OCL ob folgende Aussagen über den Systemzustand stimmen. Beginnt mit der Navigation immer bei der Mitarbeiter-Controller Instanz `m`.








- a) Alle Aufträge sind beendet. 
- b) Mindestens ein Auftrag ist noch nicht beendet. 
- c) Alle Aufträge, die keine Inspektionen sind, sind beendet.  
- d) Alle beendeten Aufträge haben einen Mitarbeiter/eine Mitarbeiterin als Beender/Beendenden vermerkt. 
- e) Alle Aufträge die einen Mitarbeiter/eine Mitarbeiterin als Beender vermerkt haben sind auch beendet. 
- f) Die IDs der Fahrzeuge sind eindeutig.  

### Lösung:

- a) `m.allOrders->forAll(a:Order | a.closed)` oder  
`m.allOrders->select(not closed)->size() = 0`
- b) `m.allOrders->exists(not closed)` oder  
`m.allOrders->select(not closed)->size() > 0` oder  
`m.allOrders.closed->count(true) > 0`
- c) `m.allOrders->select(not oclIsTypeOf(Inspection))->forAll(closed)` oder  
`m.allOrders->forAll(not oclIsTypeOf(Inspection) implies closed)`
- d) `m.allOrders->select(closed = true)->forAll(closing <> null)`
- e) `m.allOrders->select(closing <> null)->forAll(closed = true)`
- f) `m.car.id->asSet()->size() = m.car.id->size()`

## Aufgabe 6: Rekursion in OCL

Extrahiert folgende Information mit `iterate` oder `closure` aus dem Systemzustand.

- a) Wie viel Geld bringen alle Aufträge zusammen? 
- b) Wie groß ist der Anteil der Inspektionen am gesamten Umsatz in Prozent?  
- c) Erstellt einen String, in dem die Kund:innen mit den Fahrzeugtypen aller ihrer Autos aufgelistet werden.  
- d) Erstellt ein Set mit allen geraden positiven Zahlen bis 100.  

### Lösung:

- a) `m.allOrders->iterate(a;s : Integer = 0 | s + a.price)`
- b) `m.allOrders->select(oclIsTypeOf(Inspection))->iterate(i;s : Integer = 0 | s + i.price) / m.allOrders->iterate(a;s : Integer = 0 | s + a.price) * 100`
- c) `m.customer->iterate(k; s : String = '' | s + k.name + ' hat:' + k.car->iterate(f; s2 : String = '' | s2 + ' ein ' + f.typ) + '.')`
- d) `Set{0}->closure(i | if i < 100 then (i+2) else (i) endif)`



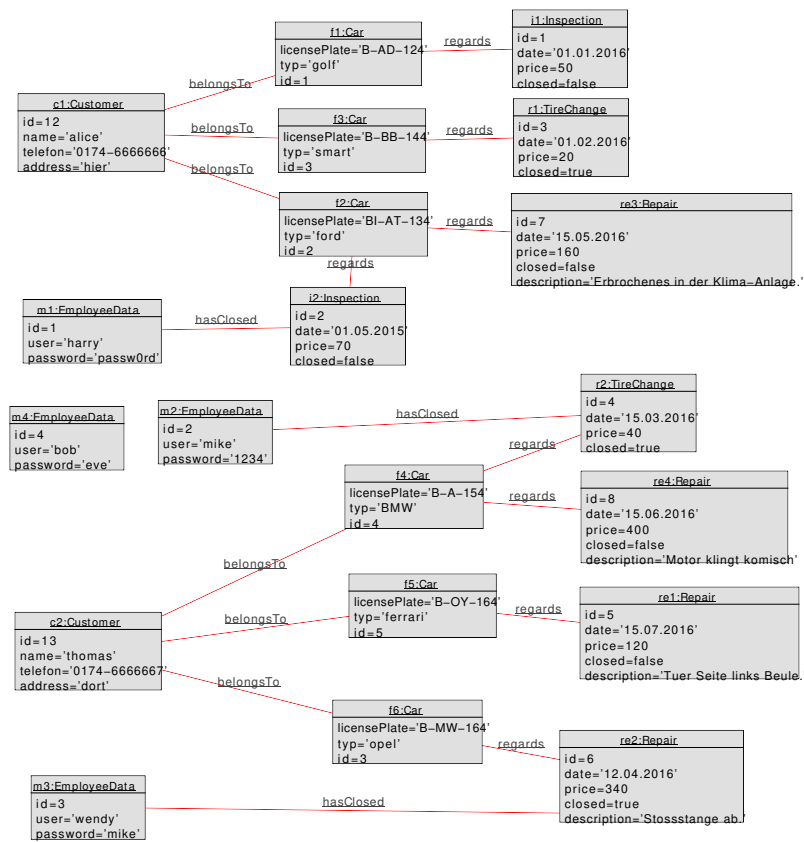


Abbildung 2: Beispiel-Objektdiagramm (nur Entity-Objekte)